Grau de Indecidibilidade da Universalidade para Subclasses de Autômatos Temporizados

Guilherme Albuquerque Pinto

Tese de Doutorado

Instituto de Computação
Universidade Estadual de Campinas

Grau de Indecidibilidade da Universalidade para Subclasses de Autômatos Temporizados

Guilherme Albuquerque Pinto

29 de agosto de 2003

Banca Examinadora:

- Prof. Dr. Arnaldo Vieira Moura
  Instituto de Computação, UNICAMP (Orientador)
- Prof. Dr. Sérgio Vale Aguiar Campos
  Departamento de Ciência da Computação, UFMG
- Prof. Dr. Arnaldo Mandel
  Instituto de Matemática e Estatística, USP
- Prof. Dr. Tomasz Kowaltowski
  Instituto de Computação, UNICAMP
- Prof. Dr. João Meidanis
  Instituto de Computação, UNICAMP
- Profa. Dra. Ana Cristina Vieira de Melo (Suplente)
  Instituto de Matemática e Estatística, USP
- Prof. Dr. Ricardo Dahab (Suplente)
  Instituto de Computação, UNICAMP

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DO IMECC DA UNICAMP

Pinto, Guilherme Albuquerque

P658g Grau de Indecidibilidade da Universalidade para Subclasses de Autômatos Temporizados / Guilherme Albuquerque Pinto – Campinas, SP: [s.n.], 2003.

Orientador: Arnaldo Vieira Moura

Tese (doutorado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Linguagens Formais. 2. Teoria dos Autômatos. 3. Funções Recursivas. I. Moura, Arnaldo Vieira. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Grau de Indecidibilidade da Universalidade para Subclasses de Autômatos Temporizados

Este exemplar corresponde à redação final da Tese devidamente corrigida e defendida por Guilherme Albuquerque Pinto e aprovada pela Banca Examinadora.

Campinas, 16 de setembro de 2003.

Prof. Dr. Arnaldo Vieira Moura
Instituto de Computação, UNICAMP
(Orientador)

Tese apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Doutor em Ciência da Computação.

TERMO DE APROVAÇÃO

Tese defendida e aprovada em 29 de agosto de 2003, pela Banca examinadora composta pelos Professores Doutores:

Prof. Dr. Sérgio Vale Aguiar Campos
DCC - UFMG

Prof. Dr. Arnaldo Mandel
IME - USP

Prof. Dr. Tomasz Kowaltowski
IC - UNICAMP

Prof. Dr. João Meidanis
IC - UNICAMP

Prof. Dr. Arnaldo Vieira Moura
IC - UNICAMP

Agradecimentos

Gostaria de agradecer ao meu orientador, Arnaldo Moura, a orientação liberal na definição dos rumos de investigação, a confiança e, principalmente, a paciência durante todos esses anos. Mais do que a orientação, devo ao Arnaldo grande parte da minha formação em teoria da computação. As três disciplinas que fiz com ele influenciaram muito nesta tese e são, até hoje, fonte de motivação.

Gostaria de agradecer ao meu orientador de mestrado, Pedro Rezende, o apoio durante o doutorado e, em especial, o incentivo por ocasião da publicação do artigo da RMU.

A todos do IC, colegas, funcionários e professores, muito obrigado por tudo. Em particular, gostaria de agradecer à Vera Ragazzi, a força em vários momentos do doutorado; e ao Mario Harada, o apoio decisivo no início de 2001.

Por fim, quero agradecer o apoio da Fundação CPqD e o suporte financeiro direto da Capes e do CNPq.

Resumo

Autômatos Temporizados são uma generalização de ω-Autômatos, e de Autômatos Finitos por conseqüência, interpretados sobre palavras infinitas temporizadas, onde a cada símbolo está associado um tempo de ocorrência. O problema da universalidade, dizer se um dado autômato aceita todas as palavras, é PSPACE-completo para autômatos determinísticos; mas é altamente indecidível quando não-determinismo irrestrito é permitido. Mais precisamente, universalidade para autômatos temporizados não-determinísticos é Π_1^1-difícil e o exato grau de indecidibilidade do problema ainda está em aberto na literatura.

Nesta tese definimos três subclasses de autômatos temporizados não-determinísticos, modificando a sintaxe dos autômatos, de modo a impor restrições ao não-determinismo e permitir a determinação do exato grau de indecidibilidade para universalidade. Para as duas primeiras classes, que são de interesse independente, mostramos que o problema da universalidade é Π_1^1-completo. Para a terceira, é Π_1^0-completo. Também estabelecemos os relacionamentos de expressividade entre as classes e mostramos que todas são subclasses próprias de autômatos temporizados não-determinísticos. Estes resultados mostram que qualquer que seja o grau de indecidibilidade do problema em aberto na literatura, ele será surpreendente do ponto de vista da expressividade das classes.

Abstract

Timed Automata are a generalization of ω-Automata, and of Finite Automata as a consequence, interpreted over infinite timed words, where each symbol is associated to an occurrence time. The universality problem, whether a given automaton accepts every word, is PSPACE-complete for deterministic automata; but becomes highly undecidable when unrestricted nondeterminism is allowed. More precisely, universality for nondeterministic timed automata is Π_1^1-hard and the exact degree of undecidability is still open in the literature.

In this thesis we define three subclasses of nondeterministic timed automata, modifying the syntax of the automata, in order to impose restrictions on the nondeterminism and allow the determination of the exact degree of undecidability for universality. For the first two, which are of independent interest, we show that the universality problem is Π_1^1-complete. For the third one, universality is Π_1^0-complete. We also establish the relationships between these classes; and show that all three are proper subclasses of nondeterministic timed automata. These results show that whatever the degree of undecidability of the open problem may be, it will be surprising from the point of view of the expressiveness of the classes.

Conteúdo

Lista de Figuras

Capítulo 1

Introdução

Esta tese trata da questão da determinação do grau de indecidibilidade para o problema da universalidade de autômatos temporizados não-determinísticos. Essa questão foi colocada em 1994 por Alur e Dill [4], no artigo em que os primeiros resultados na teoria de autômatos temporizados foram estabelecidos. Naquele artigo, o problema é demonstrado Π^1_1-difícil e os autores relatam que ele resistiu a tentativas de ser demonstrado Π^1_1-completo [4, pág. 217]. Colocando de uma maneira intuitiva, os resultados desta tese pretendem mostrar, por um lado, que não surpreende que o problema tenha resistido e, por outro lado, que qualquer que seja o grau de indecidibilidade, ele será surpreendente do ponto de vista da expressividade dos autômatos.

Definições rigorosas para autômatos temporizados, para o problema da universalidade e para as hierarquias de classificação de indecidibilidade, aparecerão no Capítulo 2. O objetivo desta introdução é, de modo geral, apresentar e discutir o problema intuitivamente; e, em particular, responder as perguntas: *por que universalidade? por que não-determinismo?* e *por que classificar problemas indecidíveis?* A intenção não é tanto justificar de forma objetiva a relevância do tema, e sim, apresentar a motivação pessoal do autor em estudar este problema. Esta introdução trará, também: uma explicação dos resultados; referências a trabalhos relacionados; e a organização do restante da tese.

Domínio do tempo. Autômatos Finitos tradicionais são interpretados sobre palavras finitas. Quando interpretamos autômatos finitos sobre palavras infinitas temos os chamados ω-Autômatos [15]. Autômatos Temporizados (*Timed Automata* [4]) são ω-autômatos interpretados sobre palavras infinitas temporizadas, onde a cada símbolo está associado

um tempo de ocorrência. Sintaticamente, o autômato possui um número finito de relógios e as transições possuem restrições para os valores de cada relógio e podem zerar o seu valor. O autômato, assim, tem algum poder para medir a distância de tempo entre símbolos da palavra.

Uma das motivações originais para a definição de autômatos temporizados foi a modelagem e verificação de sistemas de tempo-real. Neste contexto, do ponto de vista de aplicações práticas, há na literatura muita discussão sobre o domínio do tempo a ser adotado. Certos sistemas e propriedades precisariam de domínio denso, com tempos de ocorrência reais, ou racionais; para outros, bastaria um domínio discreto, com tempos de ocorrência inteiros, ou múltiplos de um certo valor racional. Algumas referências para esse debate são [26, 7, 9, 34, 17]. Do ponto de vista teórico, o domínio discreto não é muito rico, porque simplifica quase todos os problemas de decisão. A situação é mais interessante no domínio denso, que é o considerado nesta tese. Nesse caso não há uma distância mínima entre dois símbolos e uma palavra pode conter, num intervalo unitário de tempo, um número arbitrário de símbolos. Essa possibilidade é um dos fatores que dão ao autômato poder *demais*. Como veremos no Capítulo 2, um autômato temporizado não-determinístico é capaz de determinar se uma palavra temporizada *não* codifica uma computação infinita de uma máquina de Turing, algo que uma máquina de Turing não pode fazer.

Universalidade. Dois problemas de decisão são, geralmente, usados para caracterizar a complexidade computacional de uma classe de autômatos. A universalidade, dizer se um autômato aceita todas as palavras, e a não-vacuidade (*non-emptiness*), dizer se aceita pelo menos uma palavra. A universalidade, como é intuitivo, é geralmente mais complexa que a não-vacuidade. Isso é um reflexo direto da analogia entre autômatos e lógicas. O problema da universalidade em autômatos corresponde ao problema da validade em lógica, assim como não-vacuidade corresponde à satisfatibilidade. Essa analogia é discutida em [48] para ω-autômatos e em [51] para autômatos temporizados. Boa parte da teoria de autômatos surgiu, de fato, como ferramenta para solução dos problemas de decisão de lógicas. Por exemplo, dada uma fórmula da lógica em questão, obtemos um autômato equivalente, tal que o autômato é universal se e somente se a fórmula é válida.

É costume dizer que uma dada lógica é decidível, quando a validade nessa lógica é

decidível. Essa é uma das razões pelas quais o problema da universalidade é um caracterizador natural da complexidade computacional de uma classe de autômatos.

Não-determinismo. O interesse por formalismos não-determinísticos pode ser explicado de várias maneiras. No caso mais notório, a teoria de NP-completude, podemos dizer que máquinas de Turing não-determinísticas polinomiais são interessantes simplesmente porque elas caracterizam uma classe de problemas de decisão de muito interesse prático e permitem a obtenção de evidências de sua complexidade computacional relativa.

No caso do uso prático de autômatos para descrição de propriedades em verificação de programas ou sistemas, muitas vezes o formalismo determinístico e o não-determinístico caracterizam a mesma classe de propriedades ou linguagens, ou seja, são igualmente expressivos; mas pode ser muito mais simples para uma pessoa descrever certas propriedades de forma não-determinística, além de, geralmente, ser possível fazê-lo com autômatos consideravelmente menores.

Essa facilidade de descrição aparece também, em essência, naquela que consideramos ser a justificativa mais interessante para a atenção ao não-determinismo: o fato dele ser uma importante ferramenta no desenvolvimento de uma teoria. Um exemplo próximo é na teoria de ω-autômatos [15]. Demonstrar o fechamento por complementação numa classe de ω-autômatos determinísticos é trivial, porém o fechamento por projeção é complicado. Já para ω-autômatos não-determinísticos ocorre o contrário, fechamento por projeção é trivial e por complementação é complicado. O caminho mais simples para obter as duas demonstrações pode ser demonstrar a equivalência das duas classes.

Hierarquias de indecidibilidade. Como o problema da universalidade é trivialmente redutível ao problema de inclusão de linguagens—basta testar a inclusão de um autômato universal contra o autômato em questão—inclusão de linguagens também é indecidível. Em verificação de sistemas usando autômatos, a inclusão de linguagens é o principal problema de decisão: o sistema é modelado por um autômato e a propriedade a ser verificada é expressa por outro autômato. O sistema satisfaz a propriedade se a linguagem do primeiro autômato está contida na do segundo.

A impossibilidade de se encontrar um algoritmo para inclusão de linguagens para autômatos não-determinísticos traz a atenção da literatura para subclasses de autômatos temporizados, o mais expressivas possível, mas com universalidade e inclusão de lingua-

gens ainda decidíveis. Exemplos dessas subclasses são autômatos temporizados determinísticos [4], autômatos temporizados do tipo relógio de evento [5] e autômatos temporizados abertos e fechados [45]. O foco nesses trabalhos é modificar a sintaxe ou a semântica dos autômatos de modo a encontrar a fronteira da decidibilidade. A questão, porém, é que os algoritmos que resultam para essas subclasses não são, por nenhum critério razoável, eficientes na prática. Universalidade para autômatos determinísticos, por exemplo, é PSPACE-completo; os algoritmos são exponenciais ou duplamente exponenciais, e pouca esperança há de se encontrar um algoritmo eficiente para o problema, quando considerado em toda a generalidade. O maior valor desses trabalhos, na nossa opinião, para a aplicação prática de verificação, é na identificação de propriedades interessantes sobre autômatos temporizados, que eventualmente possam ser usadas na obtenção de métodos heurísticos e estruturas de dados para a implementação de ferramentas de uso prático, efetivamente eficientes para casos específicos. Não é possível argumentar que devemos prestar atenção em autômatos determinísticos porque são decidíveis e devemos esquecer os não-determinísticos porque são indecidíveis. No restante desta subseção, e na tese como um todo, pretendemos sustentar a argumentação de que a classificação de problemas indecidíveis pode ser tão boa fonte de propriedades interessantes quanto a classificação dada pela fronteira da decidibilidade.

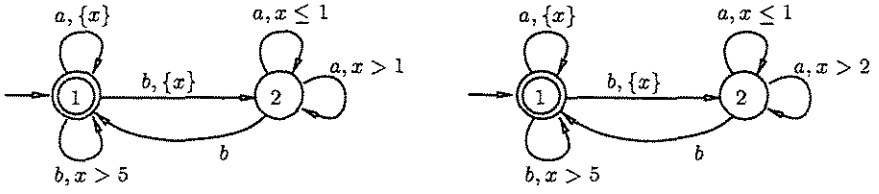

Figura 1.1: Dois autômatos: o primeiro é universal e o segundo não

Os detalhes sobre a representação dos autômatos, usada ao longo da tese, serão dados no Capítulo 2. Para a Figura 1.1, que aparece aqui com o objetivo de motivar a discussão, notamos apenas que "$\{x\}$" significa que o relógio x é zerado na transição. A figura mostra dois autômatos semelhantes: o primeiro é universal, aceita todas as palavras, e o segundo não. Uma palavra que contenha um símbolo a seguido por um b, 3 segundos depois, seguido por outro a, 1.5 segundos depois, não será aceita pelo segundo autômato.

Abrindo um parênteses, vale a pena já destacar um ponto aqui, que é absolutamente óbvio, mas, por isso mesmo, importante. Apenas a sintaxe distingue os dois autômatos e um computador, munido de um algoritmo $\mathcal{A}$, tem apenas a sintaxe do autômato como fonte de informação para determinar se ele é ou não universal. Essa observação sobre a sintaxe será, na verdade, mais importante para os capítulos seguintes. Aqui, a questão é que não pode existir tal algoritmo $\mathcal{A}$, mas ainda *é* possível classificar a complexidade do problema usando a mesma noção de algoritmo.

As hierarquias para classificação da indecidibilidade podem ser explicadas, intuitivamente, da seguinte forma. Considere, ao invés de um algoritmo $\mathcal{A}$, um algoritmo parametrizado uniformemente por um número natural, $\mathcal{A}(i)$. Esse algoritmo parametrizado é, na verdade, uma família infinita de algoritmos, $\{\mathcal{A}(1), \mathcal{A}(2), \ldots\}$. Cada um deles recebe como entrada a descrição do autômato, sempre pára, e responde *sim* ou *não*. Poderíamos agora perguntar se existe um algoritmo parametrizado $\mathcal{A}(i)$, tal que, para cada autômato, *existe* um natural n tal que $\mathcal{A}(n)$ responde *sim* se e somente se o autômato for universal. Se existisse, a universalidade estaria na classe dos problemas que admitem um algoritmo do tipo $\exists n \mathcal{A}(n)$, a classe Σ_1^0. A chamada hierarquia *aritmética* é obtida estendendo essa idéia para parametrizações por mais de um número natural e considerando o número de alternâncias de quantificadores necessário para o problema. Por exemplo: $\forall n \mathcal{A}(n)$ define a classe Π_1^0; $\exists n \forall m \mathcal{A}(n, m)$, a classe Σ_2^0; $\forall n \exists m \mathcal{A}(n, m)$, a classe Π_2^0; e assim por diante.

A hierarquia *analítica*, que será também apresentada rigorosamente no próximo capítulo, contém propriamente a hierarquia aritmética e é definida usando a mesma idéia, só que parametrizando os algoritmos por funções naturais, um recurso bem mais poderoso. Por exemplo: $\exists f \forall n \mathcal{A}(f, n)$, para função f e natural n, define a classe Σ_1^1; $\forall f \exists n \mathcal{A}(f, n)$, a classe Π_1^1; $\forall f \exists g \forall n \mathcal{A}(f, g, n)$, a classe Π_2^1; e assim por diante.

A questão é que o objeto básico na classificação pelas hierarquias de indecidibilidade continua sendo o que conhecemos por algoritmo. A diferença é que raciocinamos com famílias de algoritmos e, no caso de problemas que exigem quantificação sobre funções para serem resolvidos, raciocinamos com computação e estruturas de dados infinitas e não finitas. Apesar dessas diferenças, o exercício de encontrar, digamos, um algoritmo polinomial para um problema que sabemos ser decidível é, em essência, o mesmo do de encontrar, digamos, um algoritmo Π_1^1 para um problema que sabemos possuir um algoritmo Π_2^1.

Para finalizar, vale a pena citar a experiência do verificador de sistemas híbridos HyTech [23], que acrescenta valor à argumentação acima. Nessa tese, autômatos temporizados são vistos como generalizações de ω-autômatos. Eles, entretanto, na literatura de verificação, costumam ser mais vistos como especializações de Autômatos Híbridos [22], que são o formalismo usado no HyTech. Para autômatos temporizados, universalidade e inclusão de linguagens são indecidíveis mas não-vacuidade e alcançabilidade, que também são problemas relevantes para verificação, ainda são decidíveis (embora PSPACE-completos). Para autômatos híbridos, porém, qualquer problema de decisão não trivial, relevante para verificação, é indecidível. Os procedimentos de verificação do HyTech não garantem terminação. Quer dizer, o procedimento pode, provadamente, nunca parar. Fica a critério do usuário decidir o momento de interromper a verificação, embora, em geral, o consumo de memória seja um limitante ainda mais sério. O que ocorre é que muitos sistemas e propriedades podem ser verificados eficientemente com o HyTech. Exemplos como esses reforçam a intuição de que, para a prática de verificação, a fronteira da decidibilidade, saber o que está em $\Pi_1^0 \cap \Sigma_1^0$, não tem importância maior do que a fronteira entre Π_1^1 e Π_2^1.

1.1 Os Resultados

Na seção anterior apresentamos algumas motivações gerais para o tema da tese, dentro daquela visão tradicional na qual resultados teóricos são uma das fontes primárias de conhecimento para a obtenção de resultados práticos. Nos capítulos seguintes, porém, o desenvolvimento seguirá a motivação, somente teórica, apresentada a seguir. Com ela poderemos expor, mais propriamente, a relevância dos resultados da tese.

A Seção 2.3.1 mostra que o problema da universalidade para autômatos temporizados não-determinísticos possui um algoritmo Π_2^1 trivial. Esta é a melhor cota superior para o problema. A melhor cota inferior é Π_1^1-difícil, dada por [4]. Restam quatro possibilidades para o grau de indecidibilidade do problema, ilustradas na Figura 1.2:

1. Possui algoritmo Π_1^1 e, portanto, é Π_1^1-completo;
2. Não possui algoritmo Π_1^1, mas possui algoritmo Σ_2^1;
3. Não possui algoritmo Σ_2^1, mas também não é Π_2^1-difícil;

4. É Π_2^1-difícil e, portanto, Π_2^1-completo.

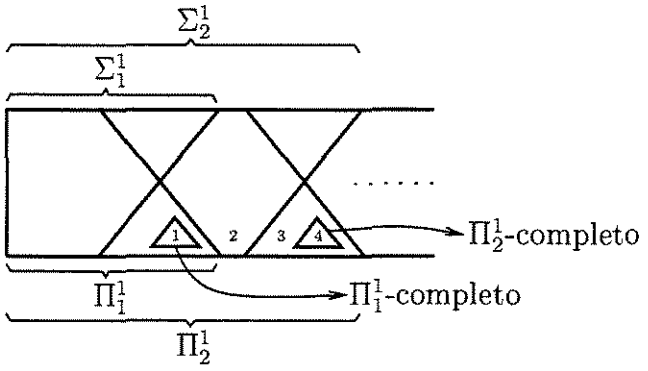

Figura 1.2: As quatro possibilidades para o grau de indecidibilidade do problema

Das quatro possibilidades, a segunda e a terceira seriam as mais surpreendentes. Isso por causa de um fenômeno [46, pág. 331] que se observa nas hierarquias: problemas definidos naturalmente, como a universalidade, por oposição a problemas mais artificiais, são geralmente completos para alguma classe das hierarquias. Por esta razão, autômatos temporizados seriam de um interesse especial caso a universalidade não fosse, provadamente, completa para nenhuma classe das hierarquias.

O outro fenômeno [46, pág. 331] que nos interessa é que, quase sempre, quando um problema é demonstrado completo para alguma classe nas hierarquias, é possível obter uma cota superior trivialmente e o complicado é demonstrar a cota inferior. Note que afirmar isso é diferente do que afirmar simplesmente que é geralmente mais complicado demonstrar cotas inferiores do que superiores. Em analogia, podemos lembrar que há algoritmos, cotas superiores, bastante complicados na literatura; mas que, geralmente, numa demonstração de NP-completude, mostrar que o problema pertence à NP é trivial e o mais complicado é a NP-dificuldade.

Quando a universalidade para autômatos temporizados, que é Π_1^1-difícil e que se espera ser completa para algum nível, resiste às primeiras tentativas de solução trivial por um algoritmo Π_1^1, surge naturalmente a suspeita de que possa ser Π_2^1-completa. No nosso caso, o problema resistiu a tentativas em todas as direções. Evidentemente, não fazemos qualquer argumentação sobre a não-trivialidade das nossas abordagens. A Seção 2.4 apresenta alguma discussão sobre as tentativas de elevar a cota inferior e de baixar a

cota superior para o problema. Em especial, a Seção 2.4.1 traz um interessante contra-exemplo para a generalização de uma tradicional construção da teoria de ω-autômatos, árvores Safra [47], na tentativa de obter um algoritmo Σ_2^1 ou Π_1^1.

A questão que esta tese responde surgiu durante a tentativa de demonstração da equivalência efetiva entre autômatos temporizados não-determinísticos e algum outro formalismo no qual fosse mais fácil demonstrar a Π_1^1-completude para universalidade. O ponto é que existe uma relação direta entre a complexidade computacional de problemas de decisão, como a universalidade, de um dado formalismo e a sua expressividade. Quer dizer, quanto mais expressivo um formalismo, mais complexo é (ou se espera ser) decidir a universalidade. Não há uma métrica que se possa usar para medir a diferença de expressividade entre dois formalismos, mas podemos, mesmo assim, colocar a pergunta: Será possível definir um formalismo, provadamente menos expressivo do que autômatos temporizados não-determinísticos, cujo problema da universalidade seja Π_1^1-completo?

Olhando a questão sobre essa ótica da expressividade podemos dizer que, daquelas quatro possibilidades, a última seria muito surpreendente porque Π_2^1-completude é o grau de indecidibilidade do problema da universalidade para dois formalismos que são provadamente mais expressivos, e intuitivamente *muito* mais expressivos, do que autômatos temporizados não-determinísticos: ω-Máquinas de Turing não-determinísticas [14] e ω-autômatos infinitos recursivos [50].

Ao contrário dos trabalhos da literatura citados na seção anterior, nossa atenção se voltou para subclasses de autômatos temporizados, o menos expressivas possível, mas com o problema da universalidade ainda Π_1^1-completo. Obtivemos duas subclasses próprias, incomparáveis por expressividade, que chamamos autômatos temporizados: *Quase Determinísticos* e *Marcadores de Passo*. Além dessas, outras duas subclasses próprias tornam a figura final do relacionamento de expressividade entre as classes, no Capítulo 5, um tanto mais interessante: autômatos temporizados *Finais* e *Zerados*. A classe de autômatos finais também é incomparável por expressividade com as duas primeiras classes, mas tem universalidade Π_1^0-completa. A classe de autômatos zerados é superclasse própria de autômatos marcadores de passo, ainda intuitivamente bem menos expressiva do que autômatos não-determinísticos, mas já resistiu à tentativa de obtenção de um algoritmo Π_1^1 para universalidade.

Com essas classes podemos ver que, do ponto de vista da expressividade, mesmo a pri-

meira daquelas quatro possibilidades, ser o problema da universalidade para autômatos temporizados não-determinísticos Π_1^1-completo, será uma situação surpreendente e interessante. Vale a pena ressaltar que a completude para as classes das hierarquias de indecidibilidade define o grau mais refinado (*1-degree*) de recursividade. Dois problemas Π_1^1-completos são isomorfos recursivamente [46]; são o mesmo problema, a menos de uma máquina de Turing. Em outras palavras, sendo o problema da universalidade para autômatos não-determinísticos Π_1^1-completo, haverá uma máquina de Turing, um *algoritmo*, que uniformemente, dado um autômato não-determinístico qualquer como entrada, produz um autômato, por exemplo, quase determinístico, que é universal se e somente se o autômato da entrada também o é. Dessa maneira, um formalismo é *mais* expressivo do que o outro mas, intuitivamente, a razão para levar em conta a diferença entre eles não poderá ser a complexidade da universalidade, que é efetivamente a mesma.

Os resultados desta tese foram publicados, de forma resumida, em [42]. Nas conclusões desta tese, Capítulo 6, retomaremos essa argumentação intuitiva. Para finalizar esta seção, listamos abaixo os teoremas que consideramos os resultados mais relevantes da tese, tanto pela importância que têm nessa argumentação, quanto pela dificuldade que nos impuseram pessoalmente:

1. Autômatos quase determinísticos são subclasse própria de autômatos não-determinísticos, Teorema 2;

2. Fechamento por interseção de autômatos temporizados marcadores de passo, Teorema 6;

3. O conceito de marcação de passo levando a um algoritmo Π_1^1, Teorema 8;

4. Se uma linguagem é aceita por um autômato marcador de passo e por um autômato final, então ela também é aceita por um autômato quase determinístico, Teorema 12.

1.2 Trabalhos Relacionados

Sobre especificamente o grau de indecidibilidade do problema da universalidade para autômatos temporizados nós não conhecemos nenhum outro trabalho, com exceção do artigo inicial de Alur e Dill [4]. Sobre o grau de indecidibilidade para outros problemas de

decisão há, de nosso conhecimento, três artigos [8, 12, 36]. Todos tratam do problema da *alcançabilidade* para generalizações de autômatos temporizados. Sobre esses trabalhos vale a pena comentar dois aspectos. O primeiro é que todos os problemas e variações estudados nesses artigos são completos *e* para alguma das quatro classes dos dois primeiros níveis da hierarquia analítica: Σ^1_1, Π^1_1, Σ^1_2 e Π^1_2. Isso confirma um terceiro fenômeno observado a respeito das hierarquias: o fato de que problemas naturais, geralmente, estão nos dois primeiros níveis [46, pág. 383].[1] O segundo aspecto é que todas as cotas superiores são triviais e merecem pouquíssima atenção nos artigos.

Sobre a teoria de autômatos temporizados, é interessante citar os trabalhos seguintes que, embora não tratem da complexidade de problemas de decisão, estudam aspectos interessantes: a caracterização através de expressões temporizadas [13]; o poder de expressividade de transições silenciosas (transições ε) [11]; lemas de iteração para autômatos sobre palavras finitas [10]; e analogias com linguagens ω-regulares [28].

1.3 Organização da Tese

O capítulo seguinte é, até a seção 2.3.2, ainda introdutório, exceto pela seção 2.3.1, que mostra como o problema pode ser facilmente localizado na classe Π^1_2. Ele apresenta a sintaxe e semântica de autômatos temporizados, introduz as hierarquias aritmética e analítica, e reproduz a demonstração de Π^1_1-dificuldade para o problema da universalidade. Em seguida, discute algumas dificuldades para localizá-lo precisamente na hierarquia e, em especial, mostra que árvores Safra não podem ser, ao menos não trivialmente, generalizadas para a obtenção de um algoritmo Π^1_1. Neste capítulo optamos, por uma questão de estilo, por não incluir uma exposição detalhada sobre as teorias de autômatos temporizados, computabilidade, hierarquias, ω-autômatos e árvores Safra.

Na seqüência, o núcleo da tese é composto por três capítulos que definem e discutem: autômatos quase determinísticos, Capítulo 3; autômatos marcadores de passo, Capítulo 4; e autômatos finais, Capítulo 5. O foco principal nesses capítulos é a determinação do grau de indecidibilidade para a universalidade e as demonstrações de que as respectivas classes são subclasses próprias de autômatos não-determinísticos. No Capítulo 3, além disso, há

[1] De fato, há quem sugira que isso possa ser explicado por uma aparente limitação da mente humana em raciocinar com duas ou mais alternâncias de quantificadores sobre estruturas infinitas.

uma seção sobre a variação da condição de aceitação e os subseqüentes efeitos na expressividade e na universalidade. Esse estudo não trouxe novidades e, por isso, não é repetido nos capítulos seguintes. Entretanto, permanece interessante como documentação. O Capítulo 5 estabelece ainda o relacionamento de expressividade entre as classes, apresentando a figura final dos resultados. Finalmente, nas conclusões, Capítulo 6, a discussão intuitiva sobre os resultados é retomada.

Capítulo 2

Autômatos Temporizados e Universalidade

Autômatos Temporizados (AT) [4] são interpretados sobre, ou aceitam, palavras temporizadas infinitas. Uma *palavra temporizada* ρ, sobre um alfabeto finito de símbolos Σ, é um par $(\overline{\sigma}, \overline{\tau})$ onde $\overline{\sigma} = \sigma_1\sigma_2\cdots$ é uma seqüência infinita de símbolos de Σ, e $\overline{\tau} = \tau_1\tau_2\cdots$ é uma seqüência *estritamente* crescente de tempos de ocorrência $\tau_i \in \mathbb{R}_{>0}$[1], satisfazendo a propriedade, ou condição, de progresso: para todo t existe j tal que $\tau_j > t$. Ocasionalmente, ao considerar uma seqüência $\overline{\tau}$, vamos nos referir ao tempo de ocorrência τ_0. Nesses casos, estaremos sempre assumindo $\tau_0 = 0$. O conjunto de todas as palavras temporizadas, ou linguagem universal, sobre um alfabeto Σ é denotado por Σ^t.

Exemplo 1 A seqüência $(a, 3.1)(b, 6)(a, 6.2)(b, 7.5)$ representa um prefixo finito de uma palavra temporizada sobre o alfabeto $\{a, b\}$. □

2.1 Autômatos Temporizados

Informalmente, AT são a generalização natural de ω-autômatos para o domínio temporal. O autômato possui um número finito de relógios, cujos valores crescem com derivada 1 com a passagem do tempo. Cada transição do autômato é anotada com uma restrição sobre os

[1] $\mathbb{R}$ e $\mathbb{Q}$ denotam os conjuntos dos números reais e racionais, respectivamente; $\mathbb{R}_{>0}$ e $\mathbb{R}_{\geq 0}$ denotam os reais maiores e maiores ou iguais a zero, respectivamente. O mesmo vale para os racionais. $\mathbb{N}$ denota o conjunto dos números naturais $\{0, 1, 2, \ldots\}$.

valores de cada relógio e uma trajetória pode tomar uma transição apenas se os relógios satisfizerem a sua restrição. Uma transição pode, ainda, zerar o valor de um conjunto de relógios após ser tomada. Nas restrições, é importante notar, o valor de um relógio pode ser comparado apenas contra constantes racionais e não contra o valor de outros relógios. Mais precisamente, dado um conjunto finito X de relógios, uma *restrição de relógio* δ sobre X é definida indutivamente pela gramática $\delta := x \leq c \,|\, x \geq c \,|\, \neg\delta_1 \,|\, \delta_1 \wedge \delta_2$, onde $x \in X$ e $c \in \mathbb{Q}_{\geq 0}$. O conjunto de todas as restrições de relógios sobre X é denotado por $\Phi(X)$. Com isso podemos definir a sintaxe básica dos autômatos temporizados: uma *tabela temporizada* é uma tupla $\mathcal{T} = \langle \Sigma, Q, Q_0, X, T \rangle$, onde

- Σ é um alfabeto finito de símbolos;
- Q é um conjunto finito de lugares;
- $Q_0 \subseteq Q$ é um conjunto de lugares iniciais;
- X é um conjunto finito de relógios;
- $T \subseteq Q \times Q \times \Sigma \times \Phi(X) \times 2^X$ é um conjunto finito de transições. Para a transição $\langle q, q', a, \delta, \lambda \rangle$ do lugar q para o lugar q' sobre o símbolo de entrada a, δ define a restrição a ser satisfeita e λ define o conjunto de relógios a serem zerados. Nós chamamos de $\mathrm{Const}(T)$, o conjunto de todas as constantes que aparecem em alguma restrição em T.

Uma *interpretação de relógio para* X é uma função de X em $\mathbb{R}_{\geq 0}$. Um *estado* de $\mathcal{T}$ tem a forma (q, ν), onde q é um lugar e ν é uma interpretação de relógio para X. Para $t \in \mathbb{R}$, nós escrevemos $\nu + t$ para a interpretação de relógio que mapeia cada relógio x em $\nu(x) + t$. Uma interpretação de relógio ν para X *satisfaz* uma restrição de relógio δ sobre X se δ é satisfeita quando cada relógio x é substituído por $\nu(x)$ em δ.

Uma *seqüência de estados* r de $\mathcal{T}$ é representada por um par $(\overline{q}, \overline{\nu})$, onde $\overline{q} = q_0 q_1 q_2 \cdots$ é uma seqüência infinita de lugares de Q e $\overline{\nu} = \nu_0 \nu_1 \nu_2 \cdots$ é uma seqüência infinita de interpretações de relógio para X. Uma seqüência de estados $r = (\overline{q}, \overline{\nu})$ de $\mathcal{T}$ é uma *trajetória de* $\mathcal{T}$ *sobre* uma palavra temporizada $\rho = (\overline{\sigma}, \overline{\tau})$ se

- $q_0 \in Q_0$, e $\nu_0(x) = 0$ para todo $x \in X$;

- para todo $i \geq 1$, existe uma transição $e = \langle q_{i-1}, q_i, \sigma_i, \delta, \lambda \rangle \in T$ tal que, $(\nu_{i-1} + \tau_i - \tau_{i-1})$ satisfaz δ, e $\nu_i(x) = 0$ se $x \in \lambda$, caso contrário $\nu_i(x) = \nu_{i-1}(x) + \tau_i - \tau_{i-1}$. Esta transição e é chamada de a i-ésima transição de r.

Um *autômato Büchi temporizado* (ABT) $\mathcal{A}$ é uma tupla $\langle \Sigma, Q, Q_0, X, T, F \rangle$, onde $\langle \Sigma, Q, Q_0, X, T \rangle$ é uma tabela temporizada e $F \subseteq Q$ é o conjunto de lugares de aceitação, ou lugares finais, de $\mathcal{A}$. Dada uma trajetória $r = (\overline{q}, \overline{\nu})$ sobre uma palavra temporizada $\rho = (\overline{\sigma}, \overline{\tau})$, seja $inf(r)$ o conjunto de lugares que ocorrem infinitas vezes em r, $inf(r) = \{s \in Q \mid \forall i \exists j (j > i, s = q_j)\}$. A trajetória r é de *aceitação* se $inf(r) \cap F \neq \emptyset$. Finalmente, a *linguagem* aceita por $\mathcal{A}$, $L(\mathcal{A})$, é o conjunto de *todas* as palavras temporizadas $\rho \in \Sigma^t$ tais que $\mathcal{A}$ possui uma trajetória de aceitação r sobre ρ. Note que, da maneira como foi definido, um ABT é não-determinístico, pode possuir incontáveis trajetórias sobre uma mesma palavra temporizada. A palavra é aceita se pelo menos uma trajetória for de aceitação. Usaremos fonte caligráfica, $\mathcal{ABT}$, para designar a classe, ou conjunto, de linguagens aceitas por algum ABT. O mesmo valerá para as subclasses de ABT definidas daqui por diante.

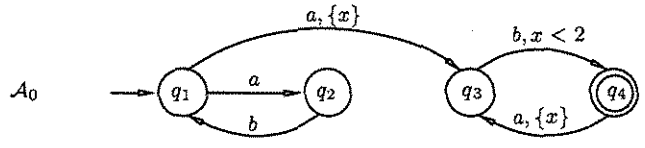

Figura 2.1: Expressando a propriedade de resposta limitada convergente com ABT

Exemplo 2 O ABT $\mathcal{A}_0$ na Figura 2.1 expressa a propriedade, ou linguagem, de resposta limitada convergente: "símbolos a e b se alternam e, a partir de um instante, a diferença de tempo entre um a e o b subseqüente é sempre menor do que 2 unidades" [4]. Nas figuras, os lugares iniciais são indicados por uma seta sem origem, os lugares de aceitação por um círculo duplo e as transições são anotadas com "σ, δ, λ", de forma que, se omitidos, $\delta = \texttt{true}$ e $\lambda = \emptyset$; se σ estiver omitido significa que há uma tal transição para cada símbolo em Σ. A linguagem aceita por $\mathcal{A}_0$ é $\{((ab)^\omega, \overline{\tau}) \mid \exists i \forall j [(j \geq i) \Rightarrow (\tau_{2j} < \tau_{2j-1} + 2)]\}$. Dado o prefixo de palavra temporizada definido no Exemplo 1, dois possíveis prefixos finitos de trajetórias de $\mathcal{A}_0$ sobre ele são: $(q_1, 0)(q_2, 3.1)(q_1, 6)(q_2, 6.2)(q_1, 7.5)$ e $(q_1, 0)(q_2, 3.1)(q_1, 6)(q_3, 0)(q_4, 1.3)$. □

Determinismo. Dada uma tabela temporizada $\mathcal{T} = \langle \Sigma, Q, Q_0, X, T \rangle$, um lugar $q \in Q$ é *determinístico* se dadas quaisquer duas transições distintas sobre o mesmo símbolo, $\langle q, q_1, a, \delta_1, \lambda_1 \rangle$ e $\langle q, q_2, a, \delta_2, \lambda_2 \rangle$ em T, $\delta_1 \wedge \delta_2$ não é satisfatível. Um conjunto $R \subseteq Q$ de lugares é *determinístico* se todos os lugares em R são determinísticos. Um ABT $\langle \Sigma, Q, Q_0, X, T, F \rangle$ é *determinístico* (ABTD) se $|Q_0| = 1$ e Q é determinístico. Essa restrição implica na propriedade de que todo ABTD tem, no máximo, uma trajetória sobre qualquer palavra. Em alguns momentos será interessante garantir que o autômato terá, pelo menos, uma trajetória sobre qualquer palavra, o que é implicado pela definição seguinte. A tabela temporizada $\mathcal{T}$ é dita *completa* se dado qualquer estado (q, ν) e qualquer símbolo $a \in \Sigma$, existe uma transição $\langle q, q_1, a, \delta_1, \lambda_1 \rangle \in T$ tal que ν satisfaz δ_1. Note que dado um ABT (ABTD) qualquer, podemos sempre obter um outro ABT (ABTD) completo, que aceita a mesma linguagem. Basta criar um novo lugar, não final, com uma transição irrestrita para si mesmo e criar, para cada lugar original e para cada símbolo, uma transição para este novo estado. Para ABT, essas transições são irrestritas; para ABTD, as restrições devem ser a negação da disjunção das restrições das transições, sobre o mesmo símbolo, do estado em questão.

Segmentos e Concatenação. Usaremos as seguintes definições naturais para segmentos e concatenação de palavras temporizadas. Dada uma palavra temporizada $\rho = (\overline{\sigma}, \overline{\tau})$:

- o seu prefixo finito de tamanho i, $i \geq 1$, é denotado por ρ_i;
- $\rho_{[i)}$, $i \geq 1$, denota a palavra temporizada que é o sufixo de ρ, da posição i em diante, com os tempos de ocorrência ajustados. Mais formalmente, $\rho_{[i)} = (\overline{\sigma'}, \overline{\tau'})$ onde $\sigma'_\ell = \sigma_{\ell+i-1}, \tau'_\ell = \tau_{\ell+i-1} - \tau_{i-1}$, $\ell \geq 1$;
- dados também $j, k > 0$, $\rho_{[j,k]}$ denota a palavra temporizada finita que é o segmento de ρ da posição j até a posição k, inclusive: $\rho_{[j,k]} = \rho'_{k-j+1}$ onde $\rho' = \rho_{[j)}$. Se $k < j$, então $\rho_{[j,k]}$ é a palavra temporizada *vazia*.

Por fim, dada uma palavra temporizada finita ρ de tamanho i e uma palavra temporizada ρ', $\rho'' = \rho \cdot \rho'$ denota a concatenação delas, quer dizer, $\rho''_i = \rho$ e $\rho''_{[i+1)} = \rho'$. Definições análogas a essas valem também para seqüências de estados finitas, segmentos e concatenação de seqüências de estados.

2.2 Hierarquias Aritmética e Analítica

Nesta seção nós abordamos sucintamente algumas definições e propriedades das hierarquias aritmética e analítica. Uma introdução abrangente sobre as hierarquias pode ser encontrada em [46]. Outras boas referências são [44, 31]. Vale a pena citar, também, uma aplicação recente e muito interessante das hierarquias na teoria de NP-completude [32].

Uma relação $H \subseteq \mathbb{N}^k$ é *recursiva* se existe uma máquina de Turing (MT) que, para qualquer entrada $\langle x_1, x_2, \ldots, x_k \rangle$, sempre pára, aceitando se e somente se $\langle x_1, x_2, \ldots, x_k \rangle \in H$. Cada nível das hierarquias possui uma classe existencial e outra universal. A primeira classe existencial da hierarquia aritmética é denotada por Σ_1^0. Um conjunto, ou problema, $S \subseteq \mathbb{N}$ pertence a Σ_1^0 se existe uma relação recursiva $H \subseteq \mathbb{N}^2$ tal que $S = \{x \in \mathbb{N} | \exists n H(n, x)\}$. Usaremos a notação simplificada $H(x_1, x_2, \ldots, x_k)$ para indicar $\langle x_1, x_2, \ldots, x_k \rangle \in H$, como nesta definição anterior. A primeira classe universal é denotada por Π_1^0. Um conjunto $S \subseteq \mathbb{N}$ pertence a Π_1^0 se existe uma relação recursiva $H \subseteq \mathbb{N}^2$ tal que $S = \{x \in \mathbb{N} | \forall n H(n, x)\}$. Com essas definições, $S \in \Pi_1^0$ se e somente se $\overline{S} \in \Sigma_1^0$, onde $\overline{S}$ é o complemento de S. As classes Σ_1^0 e Π_1^0 são, exatamente, as classes de conjuntos recursivamente enumeráveis e co-recursivamente enumeráveis, respectivamente. Sua interseção $\Delta_1^0 = \Sigma_1^0 \cap \Pi_1^0$ é a classe dos conjuntos decidíveis, ou recursivos.

Os níveis seguintes da hierarquia aritmética são obtidos considerando a alternância de quantificadores, mas não aparecerão nesta tese. Damos duas definições como exemplos. A classe existencial do segundo nível, Σ_2^0, é obtida com $\exists n \forall m H(n, m, x)$ e a classe universal do terceiro nível, Π_3^0, com $\forall n \exists m \forall o H(n, m, o, x)$.

A hierarquia analítica é definida sobre relações recursivas não apenas entre números naturais, mas entre números e funções naturais. A definição a seguir usa o conceito tradicional de MT com oráculos, que é mais simples. Entretanto, é possível definir a hierarquia com MT comuns, sem oráculos [46]. Seja $\mathbb{F}$ o conjunto de todas as funções de $\mathbb{N}$ em $\mathbb{N}$. Uma relação $H \subseteq \mathbb{F}^j \times \mathbb{N}^2$ é *recursiva* se existe uma MT com j oráculos que, para qualquer entrada $\langle x_1, x_2 \rangle$, usando oráculos $f_1, f_2, \ldots, f_j$, sempre pára, aceitando se e somente se $\langle f_1, f_2, \ldots, f_j, x_1, x_2 \rangle \in H$. Note que a MT não possui oráculos fixos, como no conceito de computação relativa a oráculos. Os oráculos, ou funções, nesse caso, são também parâmetros de entrada.

Um conjunto $S \subseteq \mathbb{N}$ pertence a Σ_1^1 se existe uma relação recursiva $H \subseteq \mathbb{F} \times \mathbb{N}^2$ tal que $S = \{x \in \mathbb{N} | \exists f \forall n H(f, n, x)\}$, onde f é uma variável de função e n uma variável

numérica. As classes Π_1^1, Σ_2^1 e Π_2^1 são definidas analogamente, com $\forall f \exists n H(f, n, x)$, $\exists f \forall g \exists n H(f, g, n, x)$ e $\forall f \exists g \forall n H(f, g, n, x)$, respectivamente, onde f e g são variáveis de função.

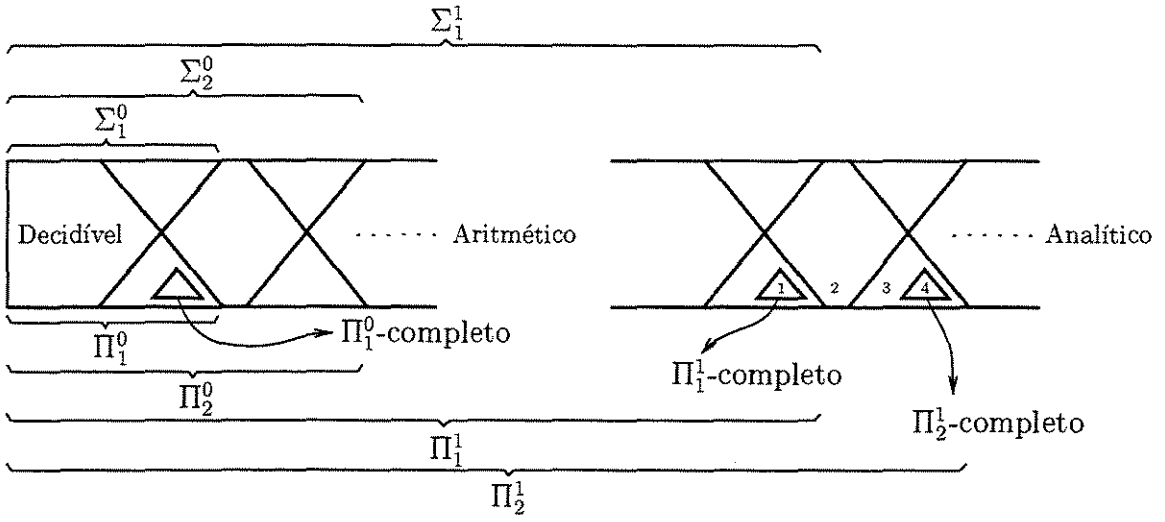

Figura 2.2: Topologia das Hierarquias Aritmética e Analítica

A Figura 2.2 apresenta a topologia das hierarquias. As seguintes propriedades, que valem também para a hierarquia analítica, merecem nota:

- Os Teoremas das Hierarquias [46] mostram que elas *não* colapsam. Para todo n, $\Sigma_n^0 \cap \Pi_n^0$ é subclasse própria de Σ_n^0 e de Π_n^0, e é superclasse própria de $\Sigma_{n-1}^0 \cup \Pi_{n-1}^0$;

- Existem conjuntos *completos* para todas as classes, segundo a definição seguinte. Dados dois conjuntos $A, B \subseteq \mathbb{N}$, A pode ser reduzido a B, $A \leq_m B$, se existe uma função recursiva $f : \mathbb{N} \to \mathbb{N}$, tal que $x \in A$ se e somente se $f(x) \in B$. Seja $\mathcal{C} \subseteq 2^{\mathbb{N}}$. Um conjunto $S \subseteq \mathbb{N}$ é $\mathcal{C}$-difícil se para todo $R \in \mathcal{C}$, $R \leq_m S$. Por fim, S é $\mathcal{C}$-completo se S é $\mathcal{C}$-difícil e $S \in \mathcal{C}$;

- Um conjunto S é Σ_n^0-completo (difícil) se e somente se $\overline{S}$ é Π_n^0-completo (difícil);

- As interseções $\Delta_n^0 = \Sigma_n^0 \cap \Pi_n^0$ não possuem conjuntos completos. Para todo $S \in \Delta_n^0$, existe $S' \in \Delta_n^0$ tal que $S' \not\leq_m S$.

2.3 Universalidade

Um ABT $\mathcal{A}$ é universal se $L(\mathcal{A}) = \Sigma^{t}$. A maneira mais natural de refinar a definição de universalidade, e que se aplica genericamente a qualquer tipo de autômato, é

Para toda palavra ρ, *existe* uma trajetória de aceitação de $\mathcal{A}$ sobre ρ. (2.1)

A seção seguinte irá mostrar como obter, trivialmente, um algoritmo Π_2^1 diretamente a partir dessa definição natural. Não se pode excluir, é claro, a possibilidade de que a universalidade tenha alguma outra definição, talvez fazendo uso de alguma propriedade específica de AT, que se revele uma melhor abordagem para a solução da questão do seu grau de indecidibilidade. Veremos, porém, que essa abordagem natural revela uma questão de fundo pelo menos tão interessante quanto a original. É sobre essa questão de fundo que, efetivamente, as subclasses definidas nos capítulos seguintes irão tratar.

2.3.1 U_{ABT} pertence a Π_2^1

Seja $\mathcal{B}_0, \mathcal{B}_1, \ldots$ uma indexação recursiva de todos os ABTs. Por isso queremos dizer: $\mathcal{B}_i$ é um ABT, para todo i; para todo ABT $\mathcal{A}$, existe i tal que $\mathcal{B}_i = \mathcal{A}$; e existe uma MT que, dado um natural n qualquer, escreve na fita uma descrição de $\mathcal{B}_n$ e pára. Note que a sintaxe de um ABT pode ser descrita finitamente sem problemas, pois todas as constantes nas restrições de relógio são racionais. A existência de uma indexação recursiva é essencial para a obtenção de cotas superiores nas hierarquias, para problemas sobre qualquer formalismo. Isso pode parecer óbvio, mas merecerá ser mais discutido nos próximos capítulos. O problema da universalidade, visto agora como um subconjunto dos naturais $U_{\text{ABT}} \subset \mathbb{N}$, pode ser refinado como

$$U_{\text{ABT}} = \{ z \in \mathbb{N} \mid \forall \rho \in \Sigma^{t} \, (\exists r \, [r \text{ é uma trajetória de aceitação de } \mathcal{B}_z \text{ sobre } \rho])\}. \quad (2.2)$$

O primeiro passo na direção de refinar essa definição até um algoritmo Π_2^1 é notar que basta considerar somente palavras temporizadas racionais. Seja $\Sigma^{\text{rt}} \subset \Sigma^{t}$ definido como $\Sigma^{\text{rt}} = \{(\overline{\sigma}, \overline{\tau}) \in \Sigma^{t} \mid \tau_i \text{ é racional, para todo } i \geq 1\}$.

Lema 1 *Seja* $\mathcal{A} = \langle \Sigma, Q, Q_0, X, T, F \rangle$ *um ABT. Dada* $\rho = (\overline{\sigma}, \overline{\tau}) \in \Sigma^{t}$, *existe* $\rho' = (\overline{\sigma'}, \overline{\tau'}) \in \Sigma^{rt}$ *tal que* $\rho \in L(\mathcal{A})$ *se e somente se* $\rho' \in L(\mathcal{A})$.

Prova. A palavra ρ' pode ser definida indutivamente exatamente como na prova do Teorema 3.17 em [4]. Com essa definição, para toda trajetória de $\mathcal{A}$ sobre ρ haverá uma trajetória de $\mathcal{A}$ sobre ρ' que segue as mesmas transições, e vice-versa. Repetimos aqui a definição para efeito de completude. Seja $\varepsilon < 1$ uma constante racional tal que para todo $c \in \{\text{Const}(T) \cup \{1\}\}$ existe um natural m, tal que $c = m\varepsilon$. Se $\tau_i = \tau_j + m\varepsilon$ para algum $0 \leq j < i$ e algum natural m, tome $\tau'_i = \tau'_j + m\varepsilon$. Caso contrário, escolha um racional τ'_i tal que $(\tau'_i - \tau'_j) < m\varepsilon$ se e somente se $(\tau_i - \tau_j) < m\varepsilon$ para todo $0 \leq j < i$ e para todo natural m. Tal escolha é sempre possível. □

Com isso, temos

$$U_{\text{ABT}} = \{\, z \in \mathbb{N} \mid \forall \rho \in \Sigma^{\text{rt}} \, (\exists\, r \, [\, r \text{ é uma trajetória de aceitação de } \mathcal{B}_z \text{ sobre } \rho\,])\} \quad (2.3)$$

e podemos agora substituir, naturalmente, o quantificador universal sobre palavras temporizadas racionais, da definição (2.3), por um quantificador universal sobre uma função natural. Para isso, seja $\Sigma = \{a_0, a_1, \ldots, a_{k-1}\}$ o alfabeto do autômato. Definimos duas funções $d_1 : \mathbb{F} \to \Sigma^\omega$ e $d_2 : \mathbb{F} \to \mathbb{Q}^\omega$ tal que: para toda $f \in \mathbb{F}$, $d_2(f)$ é uma seqüência estritamente crescente de tempos racionais (a condição de progresso será verificada no algoritmo); e dada qualquer $(\overline{\sigma}, \overline{\tau}) \in \Sigma^{\text{rt}}$, existe uma $f \in \mathbb{F}$ tal que $d_1(f) = \overline{\sigma}$ e $d_2(f) = \overline{\tau}$. A definição faz $\sigma_1 = a_{f(0) \pmod k}$, $\sigma_2 = a_{f(3) \pmod k}$, e assim por diante; e $\tau_1 = (f(1)+1)/(f(2)+1)$, $\tau_2 = \tau_1 + (f(4)+1)/(f(5)+1)$, e assim por diante. Dada $f \in \mathbb{F}$, seja $d_1(f) = \overline{\sigma}$ e $d_2(f) = \overline{\tau}$, onde para todo $i \geq 1$

- $\sigma_i = a_{f(3(i-1)) \pmod k}$, e
- $\tau_i = \tau_{i-1} + \{[(f(3(i-1)+1))+1]/[(f(3(i-1)+2))+1]\}$.

De maneira similar, para seqüências de estado do autômato, considere funções d_3 e d_4 para codificar seqüências de lugares e seqüências de interpretações de relógio.

Teorema 1 $U_{\text{ABT}} \in \Pi^1_2$.

Prova. Considere uma MT, M_{H_1}, com um oráculo. M_{H_1} aceita um dado par $\langle i, j\rangle \in \mathbb{N}^2$ se e somente se $\tau_j > i$, onde τ_j é obtido com consultas ao oráculo de acordo com d_2. Seja $H_1 \subseteq \mathbb{F} \times \mathbb{N}^2$ uma relação tal que $\langle f, i, j\rangle \in H_1$ se e somente se M_{H_1}, com oráculo f, aceita o par $\langle i, j\rangle$. Considere outra MT, M_{H_2}, com dois oráculos que, dada a tupla $\langle i, j, z\rangle \in \mathbb{N}^3$, executa a seguinte seqüência de passos:

1. Se $j \leq i$, então **rejeita**;
2. Consulta o primeiro oráculo, de acordo com d_1 e d_2, obtendo a palavra temporizada finita $\rho = (\overline{\sigma}, \overline{\tau})_j$;
3. Consulta o segundo oráculo, de acordo com d_3 e d_4, obtendo a seqüência de estados finita $r = (\overline{q}, \overline{\nu})_j$;
4. Obtém a descrição do autômato $\mathcal{B}_z = \langle \Sigma, Q, Q_0, X, T, F \rangle$, a partir de z;
5. Verifica se r é uma trajetória finita de $\mathcal{B}_z$ sobre ρ, e se $q_j \in F$. Se sim, então **aceita**; senão **rejeita**.

Tome $H_2 \subseteq \mathbb{F}^2 \times \mathbb{N}^3$ como sendo a relação tal que $\langle f, g, i, j, z \rangle \in H_2$ se e somente se M_{H_2} aceita a tupla $\langle i, j, z \rangle$, com oráculos f e g. Temos então que

$$U_{\text{ABT}} = \{z \mid \forall f \exists g \, [\, (\forall i \, \exists j \, H_1(f, i, j)) \Rightarrow (\forall i \, \exists j \, H_2(f, g, i, j, z)) \,]\} \tag{2.4}$$

Usando o algoritmo de Tarski-Kuratowski [46], podemos encontrar automaticamente uma relação recursiva H, a partir de H_1 e H_2, tal que $U_{\text{ABT}} = \{z \mid \forall f \exists g \forall n \, H(f, g, n, z)\}$. □

Apresentamos agora a questão de fundo à qual nos referimos anteriormente. Dois pontos merecem atenção na definição (2.4). O primeiro é que as variáveis de função f e g codificam, respectivamente, palavras temporizadas e trajetórias do autômato. Quer dizer, os quantificadores sobre funções no algoritmo Π^1_2 são interpretados sobre palavras e trajetórias, exatamente como na definição natural (2.1). O segundo ponto é que o predicado entre colchetes é aritmético, usa apenas quantificadores sobre números e não sobre funções. Ele afirma, dadas a palavra temporizada codificada por f e a trajetória codificada por g, que "g é de aceitação e que é mesmo sobre a palavra f, quando f respeita a condição de progresso". Na próxima seção, veremos que U_{ABT} é Π^1_1-difícil, o que significa, entre outras coisas, que o quantificador universal sobre função não pode ser evitado; quer dizer, qualquer algoritmo para o problema terá um quantificador universal sobre função. Se fixamos essa abordagem natural que interpreta a função do quantificador universal como codificadora de palavras temporizadas, a questão de saber se o problema da universalidade tem ou não um algoritmo Π^1_1 se reduz à questão de fundo: dada a palavra temporizada codificada por f, é possível afirmar, usando apenas quantificadores

sobre números, que "existe um trajetória g, que g é de aceitação e que é mesmo sobre a palavra f, quando f respeita a condição de progresso"? É ou não possível prescindir do quantificador existencial sobre função?

Essa questão merece, também, ser elaborada à luz da discussão intuitiva ensejada na Introdução. Um algoritmo parametrizado, dada a palavra codificada por f, a cada vez que é executado, dá um número finito de passos e pára. É claro, a cada execução ele pode, consultando o oráculo, "enxergar" apenas um prefixo finito da palavra. Ele pode, também obviamente, simular todas as possíveis trajetórias finitas do autômato sobre o prefixo finito da palavra que enxerga. A questão, então, é saber se existe algum conjunto de informações finito que, sendo produzido pelo algoritmo a cada execução a partir do que ele enxerga, possa, através de quantificação sobre números, afirmar que "existe uma trajetória g de aceitação sobre f". Em outras palavras, é possível afirmar que existe a trajetória *infinita* a partir de algum conjunto de informações *finito*? Árvores Safra são, justamente, uma estrutura de dados finita que responde afirmativamente a essa questão para ω-autômatos, mas que falha para ABTs, como será visto na seção 2.4.1. Também, os capítulos 3 e 4, do ponto de vista dessa discussão, mostram é possível obter restrições sintáticas simples sobre a definição de ABT que impõem restrições às trajetórias dos autômatos, de forma que é possível obter facilmente tal estrutura de dados finita. Mas essa facilidade, nesses casos, traz consigo a redução da expressividade dos autômatos.

2.3.2 U_{ABT} é Π^1_1-difícil

Em [19], o problema de decidir se uma MT não-determinística possui uma computação infinita, sobre a fita vazia, que visita o estado inicial da máquina infinitas vezes, é demonstrado Σ^1_1-completo. No artigo de Alur e Dill [4], o *complemento* de um problema equivalente a esse é reduzido à universalidade de ABTs, estabelecendo que U_{ABT} é Π^1_1-difícil. Vamos rever essa redução.

Uma MT de 2 contadores não-determinística M consiste de uma seqüência de k instruções e dois contadores, C e D. Há 6 tipos de instruções (não é necessário, nesse caso, uma instrução de parada): (a) incrementar C e pular não-deterministicamente para instrução x ou y; (b) decrementar C e pular não-deterministicamente para instrução x ou y; (c) se $C = 0$ pular para a instrução x, caso contrário pular para instrução y; (d), (e) e (f) são análogas às anteriores, trocando C por D. Uma *configuração* de M é uma tupla

(i, c, d), onde c e d são os valores correntes dos contadores, e i é a instrução a ser executada. Uma *computação* de M é definida, de forma tradicional, como uma seqüência de configurações consecutivamente relacionadas começando com $(1, 0, 0)$. Uma computação é *recorrente* se a instrução 1 é executada infinitas vezes. Defina a linguagem temporizada L sobre o alfabeto $\{b_1, b_2, \ldots, b_k, a_1, a_2\}$ tal que $(\overline{\sigma}, \overline{\tau}) \in L$ se e somente se:

1. $\overline{\sigma} = b_{i_1} a_1^{c_1} a_2^{d_1} b_{i_2} a_1^{c_2} a_2^{d_2} \ldots$, onde $(i_1, c_1, d_1)(i_2, c_2, d_2) \ldots$ é uma computação recorrente de M;

2. Para todo $j \geq 1$:

   (a) b_{i_j} ocorre no tempo j;

   (b) se $c_{j+1} = c_j$, então para todo a_1 no tempo $t \in (j, j+1)$ existe um a_1 no tempo $t + 1$;

   (c) se $c_{j+1} = c_j + 1$, então para todo a_1 no tempo $t \in (j+1, j+2)$ existe um a_1 no tempo $t - 1$, exceto para o último a_1;

   (d) se $c_{j+1} = c_j \dot{-} 1$, então para todo a_1 no tempo $t \in (j, j+1)$ existe um a_1 no tempo $t + 1$, exceto para o último a_1;

   (e) as mesmas condições de (b), (c) e (d), trocando a_1 por a_2, e c por d.

A linguagem L não pode ser aceita por um ABT, mas a linguagem complementar, $\overline{L}$, sim. Ela é uma disjunção de várias linguagens temporizadas simples, que podem todas ser aceitas por ABTs[2]. A união disjunta de todos esses ABTs é universal se e somente se M *não* possui uma computação recorrente [4].

2.4 Direções de Investigação

Para completar a revisão de resultados da teoria de ABTs faltou abordar as propriedades de fechamento das duas classes já vistas. As classes $\mathcal{ABT}$ e $\mathcal{ABTD}$ são fechadas por união e interseção, mas *não* por complemento [4]. Para $\mathcal{ABTD}$, os problemas de não-vacuidade, universalidade e inclusão de linguagens são PSPACE-completos [4]. Para $\mathcal{ABT}$, como vimos, não-vacuidade é PSPACE-completo, mas universalidade e inclusão de linguagens

[2]Na seção 3.3 serão dados exemplos dessas linguagens e seus respectivos autômatos, para o caso quase determinístico.

são indecidíveis, estão em Π_2^1 e são Π_1^1-difíceis. A Figura 2.3 ilustra o relacionamento de expressividade entre essas classes, trazendo ainda a complexidade da universalidade. Ela aparece aqui para efeito de comparação com a figura final no Capítulo 5.

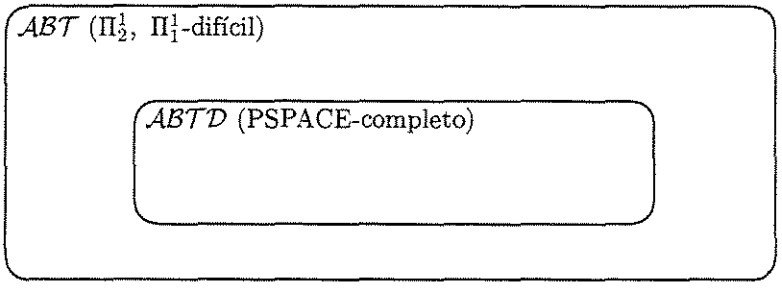

Figura 2.3: Relacionamento entre as classes de ABT

As quatro possibilidades, já mencionadas na Introdução, para o grau de indecidibilidade de U_{ABT}, estão indicadas na Figura 2.2:

1. Possui algoritmo Π_1^1; portanto é Π_1^1-completo;
2. Não possui algoritmo Π_1^1, mas possui algoritmo Σ_2^1;
3. Não possui algoritmo Σ_2^1, mas também não é Π_2^1-difícil;
4. É Π_2^1-difícil; portanto é Π_2^1-completo.

O problema está provadamente dentro de Π_2^1 e fora de Σ_1^1 (por ser Π_1^1-difícil). A seguir apresentamos alguma discussão, novamente ainda intuitiva, sobre algumas abordagens na tentativa de baixar a cota superior ou elevar a cota inferior do problema. A seção 2.4.1, entretanto, apresentará mais rigorosamente uma dessas abordagens.

Um ABT não é capaz de simular uma computação de uma MT de dois contadores. Para fazer isso, ele precisaria, por exemplo, de relógios com derivadas distintas e possibilidade de comparação de valores entre relógios [24]. Note que na redução da seção 2.3.2 o autômato é universal se *não* existe uma computação de uma certa MT. Ele aceita toda palavra que *não* codifica uma computação da MT. Mas, essencialmente, a palavra temporizada é que traz a computação da MT. O aparente poder de simulação do autômato vêm, na verdade, da palavra. De uma certa forma, no que se refere a simulação da computação

da MT, o autômato é passivo na redução. Isso também pode ser colocado da seguinte forma. A complexidade que o conjunto de trajetórias não-determinísticas do autômato apresenta, sobre uma dada palavra, vem da complexidade da palavra e não da aparente capacidade de computação do autômato por seus próprios meios. Essa observação é importante numa tentativa de se obter uma redução similar à da seção 2.3.2, mas para Π_2^1-dificuldade, como discutido a seguir.

Nos artigos [14, 50], onde é demonstrada a Π_2^1-dificuldade da universalidade para ω-MT não-determinísticas e ω-autômatos infinitos recursivos, a abordagem não é por redução a partir de outro problema já Π_2^1-difícil, mas diretamente por uma redução genérica a partir de um problema Π_2^1 qualquer. Se um conjunto está em Π_2^1, então ele possui um algoritmo, ou definição, do tipo $\{x \in \mathbb{N} | \forall f \exists g \forall n H(f, g, n, x)\}$. Naquelas reduções, a função f é codificada na entrada, ou melhor, pela palavra infinita no caso de ω-autômatos, e pelo conteúdo da fita infinita de entrada no caso de ω-MT. A entrada é reservada exclusivamente para codificar a função f. Às computações não-determinísticas da ω-MT e às trajetórias não-determinísticas do ω-autômato, cabe o papel de codificar, ou representar, a função g. O algoritmo funciona, intuitivamente, da seguinte forma. Dada a fita de entrada, a ω-MT simula todas as possíveis computações da MT M_H (a MT subjacente à relação recursiva H) na tentativa de identificar a existência da função g. A simulação é feita de forma parametrizada, como já foi discutido, mas o fato é que a informação da fita de entrada apenas codifica f e não é usada para estimular a simulação realizada pela ω-MT. De uma outra maneira, nessa abordagem de redução, a entrada deve estar livre para codificar a função f, e a existência da função g tem que ser verificada pela simulação de M_H, pelo formalismo em questão, *por seus próprios meios.* Por essa razão, definitivamente, uma generalização trivial dessas idéias para o caso de ABTs está descartada. Evidentemente, essa não é a única abordagem para tentar mostrar a Π_2^1-dificuldade de U_{ABT}, porém podemos considerar, sempre intuitivamente, que qualquer outra solução seria, em algum sentido computacional, equivalente à essa.

Soa bem mais simples uma tentativa de demonstrar a Σ_1^1-dificuldade de U_{ABT}, reduzindo a partir de conjuntos do tipo $\{x \in \mathbb{N} | \exists f \forall n H(f, n, x)\}$. Isso seria uma maneira de demonstrar que o problema não pode possuir algoritmo Π_1^1. Em comparação com a redução da Seção 2.3.2, precisaríamos obter um ABT que fosse universal se e somente se aquela MT *possui* uma computação recorrente. O que parece dificultar essa tentativa é

uma obviedade, a natureza "universal" da quantificação na definição do problema da universalidade. Depois do problema resistir a diversas tentativas, ficou a seguinte impressão: se o problema pudesse ser demonstrado Σ_1^1-difícil, poderia também ser demonstrado Π_2^1-difícil; se fosse possível obter um algoritmo Σ_2^1, seria possível também obter um algoritmo Π_1^1. Nos dois casos, o quantificador existencial parece inócuo para U_{ABT}.

Os capítulos seguintes suportam a argumentação de que seja qual for o grau de indecidibilidade de U_{ABT}, ele será surpreendente do ponto de vista da expressividade temporal. A direção de investigação foi definir subclasses de $\mathcal{ABT}$ e não superclasses, o que também poderia trazer propriedades interessantes sobre ABTs e nova luz sobre o problema. A razão da escolha é a conjectura que fazemos:

Conjectura 1 *Se U_{ABT} é completo para alguma classe, então é Π_1^1-completo.*

2.4.1 Árvores Safra e Autômatos Temporizados

Uma leitura desta seção pode ajudar o entendimento da dificuldade envolvida na questão central da tese, e pode ajudar a colocar em perspectiva os resultados dos capítulos seguintes. Porém, o que será visto aqui não será usado naqueles capítulos; seu maior valor é como documentação.

Árvores Safra são uma construção usada para obter um ω-autômato determinístico (do tipo Rabin) equivalente a um dado ω-autômato não-determinístico do tipo Büchi. A definição da árvore, que não é muito simples, e uma excelente explicação intuitiva do seu funcionamento, podem ser encontradas em [47, 48]. A definição da árvore para ABT, que consideramos, é a mesma que a usada em [47, 48], igualando o que é chamado de "state" naqueles artigos ao que chamamos de estado aqui. Nesta seção nos limitaremos a apresentar um contra-exemplo, um ABT e uma palavra temporizada, tal que o algoritmo Π_1^1 natural usando árvores Safra falha. Apontaremos a propriedade que leva à falha, sem contudo entrar nos detalhes da definição da árvore que, na verdade, não são importantes; daremos apenas alguma intuição sobre as árvores.

Cada estado do ω-autômato obtido na construção é uma árvore Safra numa forma normal. Cada nó de uma árvore Safra é um subconjunto dos estados do ω-autômato original, e pode estar colorido de branco ou de verde. O autômato obtido tem um número finito de estados, essencialmente, porque a altura da árvore é limitada, na definição, pelo

número de estados do autômato original, que é finito. No autômato determinístico obtido, cada palavra possui apenas uma trajetória, que podemos imaginar como sendo uma seqüência infinita de árvores Safra. A trajetória é de aceitação se existe um nó, chamado de recorrente, que é colorido de verde infinitas vezes na seqüência. Um algoritmo Π^1_1 para U_{ABT}, usando árvore Safra, não precisaria construir um novo autômato. O que ele aproveitaria da construção é a função de transição que transforma uma árvore em outra, dado o símbolo seguinte e seu tempo de ocorrência. O algoritmo teria a forma $\{x \in \mathbb{N} \mid \forall f\,[\,(\forall i \exists j H_1(f,i,j)) \Rightarrow (\exists p \forall i \exists j H(f,p,i,j,x))\,]\,\}$, onde f codifica a palavra temporizada, o natural p indica um nó da árvore e a MT M_H aceita se $j > i$ e a j-ésima árvore de Safra de $\mathcal{B}_x$ sobre a palavra codificada por f possui o nó p colorido de verde.

A árvore é definida de forma que o limite na sua altura implica na existência de tal nó recorrente. No caso de ABT, o espaço de estados é incontável, portanto não há em princípio um limite para a altura da árvore e nem, conseqüentemente, a garantia da existência de um nó recorrente. Entretanto, isso é um resultado a se demonstrar. De fato, para que não haja um nó recorrente, um ramo da árvore tem que crescer indefinidamente, sendo infinito no limite. A intuição é que a seqüência de árvores Safra procura ir podando as ramificações nas trajetórias de aceitação para tentar chegar a uma situação em que as trajetórias que restam "parecem" ser todas de aceitação. Esse é o momento em que um nó é colorido de verde. Esse procedimento é repetido recursivamente para todos os nós.

Apresentamos agora a propriedade que leva à falha. Seja $A = \langle \Sigma, Q, Q_0, X, T, F \rangle$ um ABT e ρ uma palavra temporizada. Primeiro notamos que dois estados distintos (q, ν) e (q', ν') são considerados *equivalentes*, se $q = q'$ e para todo $x \in X$, $\nu(x) = \nu'(x)$ *ou* $\nu(x) > \mathbf{b}$ e $\nu'(x) > \mathbf{b}$, onde $\mathbf{b}$ é a maior das constantes em $\text{Const}(T)$. O autômato não é capaz de diferenciar dois estados equivalentes. A propriedade é a seguinte:

- Toda trajetória de aceitação $r = (\overline{q}, \overline{\nu})$ de A sobre ρ é tal que:
  - A trajetória r bifurca infinitas vezes, pelo não-determinismo de A, em trajetórias que não são de aceitação; quer dizer, para todo i, existe $j > i$ tal que existe uma trajetória r^j, que *não* é de aceitação, e que é igual a r até a posição j, ou seja, $r^j_j = r_j$;
  - Essas trajetórias são todas disjuntas (nunca duas confluem); quer dizer, se $o \neq p$, para todo k, $k > o$ e $k > p$, o estado na k-ésima posição da trajetória

r^o não é equivalente ao da trajetória r^p.

Dizemos que uma trajetória $r = (\overline{q}, \overline{\nu})$, no instante (ou posição) i, *está* no lugar ℓ, se $q_i = \ell$. O fato de que o espaço de estados de um ABT é incontável não é suficiente para mostrar facilmente que essa propriedade anterior pode ser satisfeita. Para ver isso, notamos que o número de lugares e de transições no ABT é finito, de forma que a propriedade implica que para todo i, haverá um $j > i$, tal que para todo $k > j$, haverá algum lugar ℓ_k tal que *pelo menos* i trajetórias estarão dentro de ℓ_k no instante k. Para que essas trajetórias não confluam, os relógios precisam manter valores menores ou iguais à **b**. Para isso, como o tempo diverge, é preciso haver transições que zeram relógios, e essas transições, intuitivamente, não podem ser tomadas simultaneamente por duas trajetórias, caso contrário as trajetórias acabarão confluindo em algum instante posterior. A questão é a seguinte: é possível garantir que duas trajetórias nunca irão confluir, sabendo que o número de trajetórias aumenta monotonicamente e sem limite, que toda trajetória precisa tomar, infinitas vezes, transições que zeram relógios, e que o número de transições, relógios e lugares é finito?

Por algum tempo julgamos que o algoritmo poderia funcionar, porque aquela propriedade não pode ser satisfeita em algumas situações. Por exemplo, se o alfabeto possui apenas um símbolo. Entretanto é possível encontrar um contra-exemplo, apresentado a seguir, com dois símbolos, um relógio, dois lugares e cinco transições.

O Contra-exemplo

A Figura 2.4 apresenta o autômato $\mathcal{A}_{\text{safra}}$. Listamos abaixo algumas propriedades desse autômato que podem ser verificadas sem dificuldade. Em seguida definiremos a palavra temporizada $\rho = (\overline{\sigma}, \overline{\tau})$ que completa o contra-exemplo.

- $\mathcal{A}_{\text{safra}}$ é universal. Toda palavra possui trajetória de aceitação pela transição do lugar 1;
- $\mathcal{A}_{\text{safra}}$ é completo. Nenhuma trajetória termina por impossibilidade de transitar;
- O lugar 2 é determinístico, portanto nenhuma nova trajetória é criada no lugar 2;
- A única transição que zera o relógio x, no lugar 2, é a da direita; portanto a única maneira de duas trajetórias confluírem, se tornando uma só, é tomando essa transição

no mesmo instante. Note, também, que uma trajetória, no instante que transita do lugar 1 para o lugar 2, não pode confluir com alguma trajetória que já esteja no lugar 2, porque ela chega com x zerado e todas as outras tomam, no mesmo instante, a transição de baixo, que não zera x.

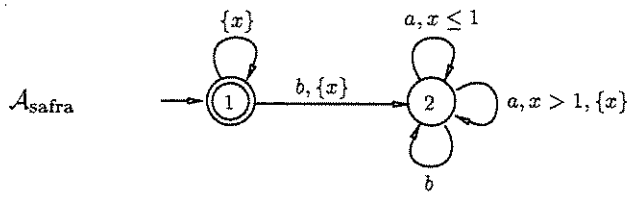

Figura 2.4: Contra-exemplo para o algoritmo Π_1^1 baseado em árvores Safra

A seqüência de tempos de ocorrência $\overline{\tau}$ é dada pela série harmônica tradicional, ou seja, $\tau_i = \tau_{i-1} + \frac{1}{i}$, que é divergente. A seqüência de símbolos $\overline{\sigma}$ é do tipo $(a^+b)^\omega$ e os primeiros dez símbolos são *ababaaabaa*. A Figura 2.5 ilustra as trajetórias finitas de $\mathcal{A}_{\text{safra}}$ sobre $\rho_{[1,10]}$. Após o décimo símbolo, as três trajetórias no lugar 2 possuem os seguintes valores aproximados para o relógio x: 0,211111; 0,845635 e 0,336111.

Uma nova trajetória será criada a cada vez que ocorrer um símbolo b. O objetivo é definir o restante de ρ de forma que nenhum par de trajetórias tome a transição da direita simultaneamente. O primeiro-fato importante a observar é garantido pela série harmônica. Como o relógio só é zerado no instante em que ocorre um símbolo, a diferença entre os valores de x para duas trajetórias distintas no lugar 2 é sempre *maior* do que o tempo até a ocorrência do próximo símbolo. Com isso, após um símbolo b, podemos adicionar quantos símbolos a forem necessários antes de escolhermos adicionar o próximo b, porque há a garantia de que somente uma trajetória por vez poderá tomar a transição da direita (somente uma trajetória terá seu relógio com valor maior do que 1).

Falta ver que sempre será possível adicionar o próximo b. A adição de um b faz todas as trajetórias no lugar 2 tomarem a transição de baixo, que não zera os relógios. O risco é que, no a seguinte, duas trajetórias poderiam ser forçadas a tomar a transição da direita. Porém, note que se há $k-1$ trajetórias distintas no lugar 2, se ordenarmos as trajetórias pelo valor do relógio x, haverá sempre pelo menos um intervalo de tempo, entre os valores de x na ordenação, de tamanho pelo menos $\frac{1}{k}$. Dessa forma, primeiro adicionamos a's

até que a diferença de tempo entre os símbolos da palavra seja menor do que $\frac{1}{2k}$, o que a série harmônica possibilita. Depois adicionamos a's até que a trajetória cujo valor de x definia o limite superior daquele intervalo mencionado seja aquela com o maior valor de x. Feito isso, o próximo símbolo pode ser um b que, garantidamente, apenas uma trajetória tomará a transição da direita no a seguinte.

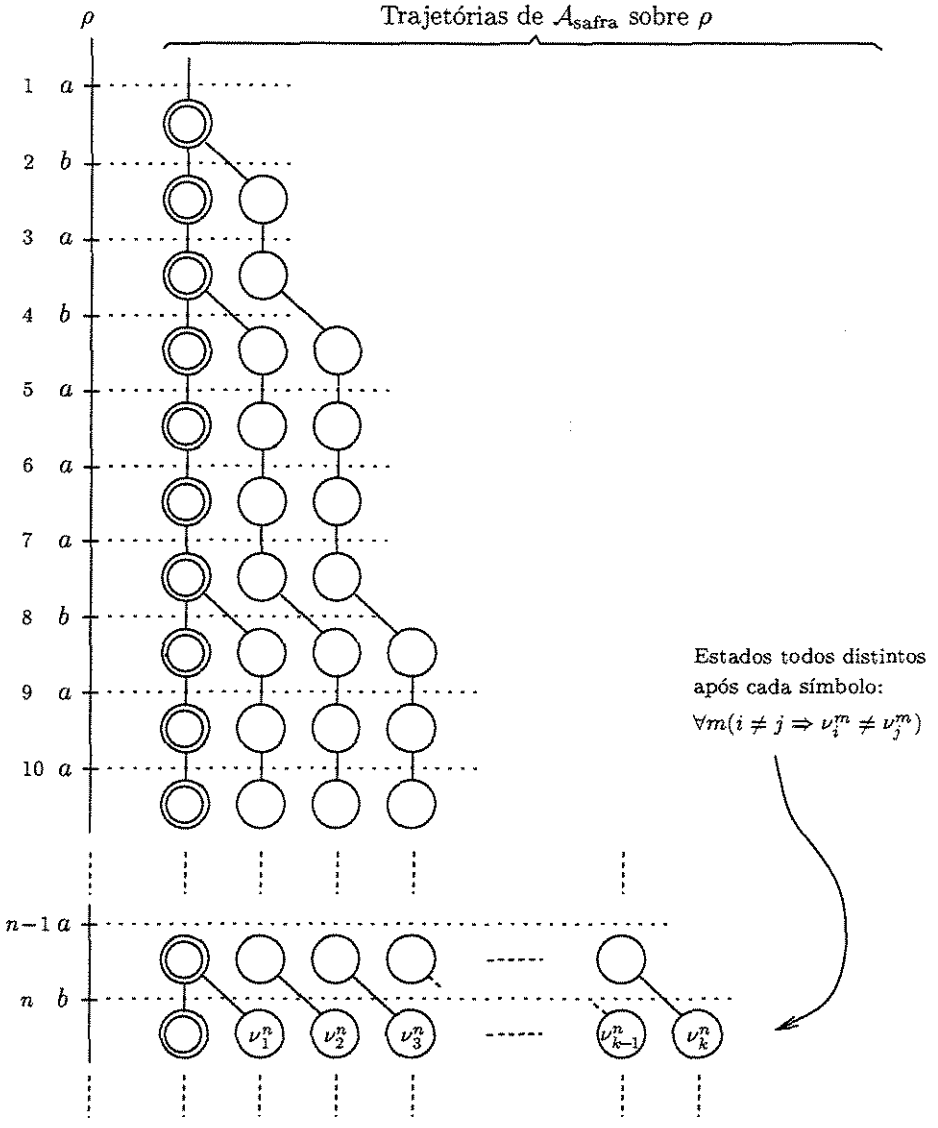

Figura 2.5: Trajetórias de $\mathcal{A}_{\text{safra}}$ sobre ρ

Restrição Sintática. Note que neste autômato $\mathcal{A}_{\text{safra}}$ o lugar 2 e suas transições são inúteis, ou desnecessários; nenhuma trajetória de aceitação passa por ele. A mesma linguagem, Σ^t, seria aceita se o removêssemos. Chamemos informalmente esse lugar e suas transições, nesse caso, de elementos de sintaxe inúteis. A pergunta que surge disso é a seguinte. A propriedade que leva à falha do algoritmo baseado em árvores Safra pode ser satisfeita somente quando o autômato possui elementos de sintaxe inúteis? De outra maneira, seria possível satisfazer àquela propriedade obrigando que cada uma das infinitas trajetórias que bifurcam da trajetória de aceitação passem infinitas vezes por elementos de sintaxe *da trajetória de aceitação*?

Não foi possível responder afirmativamente à nenhuma das duas perguntas. Note que se elementos de sintaxe inúteis fossem necessários, então existiria um algoritmo Π^1_1 da forma $\{x \in \mathbb{N} | \forall f\,[(\forall i \exists j H_1(f,i,j)) \Rightarrow (\exists p \exists r \exists m \forall i \exists j H(f,p,r,m,i,j,x))]\}$, onde r codificaria os elementos de sintaxe (lugares, transições ou mesmo apenas certas restrições de relógio) que o algoritmo passaria a desconsiderar, a partir do instante m, na simulação das árvores Safra de $\mathcal{B}_x$ sobre a palavra codificada por f.

Capítulo 3

Autômatos Quase Determinísticos

Neste capítulo apresentaremos autômatos temporizados quase determinísticos. A propriedade essencial de autômatos quase determinísticos, que se aplica genericamente a qualquer tipo de ω-autômato, é a seguinte:

Propriedade 1 Se uma trajetória do autômato é de aceitação, então ela contém um número *finito* de transições não-determinísticas.

O conceito de quase determinismo foi introduzido na teoria de ω-autômatos [16, 47] como uma maneira de se obter melhores cotas superiores para um problema de verificação probabilística, onde um passo dos algoritmos é a construção de um autômato determinístico equivalente a um não-determinístico. Para ω-autômatos, é possível obter um autômato equivalente, quase determinístico, muito menor do que o determinístico, e resolver a verificação probabilística sobre ele com custo menor. Por causa dessa equivalência, do ponto de vista de expressividade, quase determinismo não é muito rico para ω-autômatos. A situação é mais interessante para ABT. Na seção 3.1 mostraremos que a classe de linguagens quase determinísticas é subclasse própria de $\mathcal{ABT}$. Depois disso, veremos que podemos obter um algoritmo Π_1^1 para essa classe que ainda é Π_1^1-difícil.

A propriedade 1 pode ser forçada com a restrição sintática natural seguinte. Para uma tabela temporizada $\langle \Sigma, Q, Q_0, X, T \rangle$ e um conjunto $S \subseteq Q$, seja $\mathrm{Reach}(S) \subseteq Q$ o conjunto de todos os lugares $s \in Q$ para os quais existe uma seqüência $s_1, s_2, \ldots, s_k$, $k \geq 1$, tal que $s_1 \in S$, $s_k = s$ e para todo $1 \leq i < k$ existe $\langle s_i, s_{i+1}, a, \delta, \lambda \rangle$ em T. Um ABT $\langle \Sigma, Q, Q_0, X, T, F \rangle$ é *quase determinístico* (ABTQD) se $\mathrm{Reach}(F)$ é determinístico.

Dois exemplos de ABTQDs são os autômatos $\mathcal{A}_0$ e $\mathcal{A}_1$ das Figuras 2.1 e 3.1. O ABT $\mathcal{A}_2$ da Figura 3.1 *não* é um ABTQD.

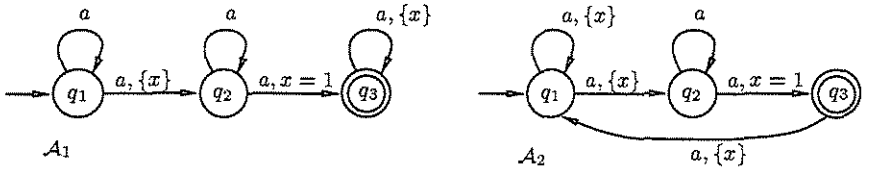

Figura 3.1: Exemplo de quase determinismo e não-determinismo

Impor restrições sintáticas simples à definição de ABT tem a vantagem de trazer, automaticamente, dois resultados desejáveis para nossa argumentação: a classe de linguagens aceitas pelos autômatos restritos é uma subclasse de $\mathcal{ABT}$; e fica garantida a existência de uma indexação recursiva para os autômatos restritos, o que será discutido mais a frente.

Propriedades de Fechamento. A classe $\mathcal{ABTQD}$ possui a mesma robustez de $\mathcal{ABT}$ em relação a propriedades de fechamento. Ela é fechada por união e interseção, mas não por complemento. Dado um conjunto $\{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k\}$ de ABTQDs, $\bigcup_{i=1}^{k} L(\mathcal{C}_i)$ é aceita pela união disjunta de $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k$, que é um ABTQD. Para $\bigcap_{i=1}^{k} L(\mathcal{C}_i)$, nós notamos simplesmente que a construção do produto de [4, p. 197], usada para mostrar o fechamento por interseção para $\mathcal{ABT}$, quando aplicada sobre $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k$, gera outro ABTQD. Para o não-fechamento por complementação, notamos que o complemento da linguagem $L(\mathcal{A}_1)$ (Figura 3.1) não pode ser aceito sequer por um ABT. A demonstração é similar à do Teorema 2. Intuitivamente, um ABT não pode afirmar que *não* existe um par de símbolos separados por uma distância de exatamente uma unidade de tempo, porque possui um número finito de relógios e pode haver qualquer número de símbolos num intervalo de uma unidade de tempo. Veja discussão relacionada em [4] e uma prova alternativa para isso em [30].

Em [4] o não-fechamento por complementação de $\mathcal{ABT}$ é obtido com uma demonstração diferente, que vale a pena ser repetida aqui como exemplo do uso da classificação de problemas indecidíveis no desenvolvimento de uma teoria. A Π_1^1-dificuldade de U_{ABT} implica na Π_1^1-dificuldade do problema da inclusão de linguagens. Dados ABTs $\mathcal{C}_1$ e $\mathcal{C}_2$, $L(\mathcal{C}_1) \subseteq L(\mathcal{C}_2)$ se e somente se $L(\mathcal{C}_1) \cap \overline{L(\mathcal{C}_2)} = \emptyset$. Suponha que $\mathcal{ABT}$ *é* fechada por complementação. Então, podemos mostrar facilmente que $L(\mathcal{C}_1) \not\subseteq L(\mathcal{C}_2)$ se e somente

se existe $\mathcal{A}$ tal que $L(\mathcal{C}_1) \cap L(\mathcal{A}) \neq \emptyset$ e $L(\mathcal{C}_2) \cap L(\mathcal{A}) = \emptyset$, tomando $\mathcal{A}$ como o autômato que aceita $\overline{L(\mathcal{C}_2)}$. Mas então, como $\mathcal{ABT}$ é fechada por interseção e não-vacuidade é decidível, e como o conjunto de todos os ABTs tem uma indexação recursiva $\mathcal{B}_0, \mathcal{B}_1, \ldots$, o complemento do problema da inclusão de linguagens seria recursivamente enumerável, estaria em Σ_1^0: $\exists k[L(\mathcal{C}_1) \cap L(\mathcal{B}_k) \neq \emptyset \wedge L(\mathcal{C}_2) \cap L(\mathcal{B}_k) = \emptyset]$. Mas isso é uma contradição, porque o complemento de um problema Π_1^1-difícil não pode estar em Σ_1^0.

3.1 Expressividade

O ABTQD $\mathcal{A}_1$, na Figura 3.1, aceita toda palavra temporizada sobre $\{a\}$ tal que existe um par de a's separados por uma diferença de exatamente uma unidade de tempo, quer dizer, $L(\mathcal{A}_1) = \{(a^\omega, \overline{\tau}) \mid \exists i \exists j[(i < j) \wedge (\tau_j = \tau_i + 1)]\}$. Embora $L(\mathcal{A}_1)$ possa ser aceita por um ABTQD, ela não pode ser aceita por um ABTD [4]. A razão intuitiva também é que, como o número de a's que podem ocorrer num intervalo unitário é ilimitado, um ABTD precisaria de um número ilimitado de relógios para reconhecer corretamente tal par de a's. Como um ABTD não pode reconhecer um tal par, um ABTQD não deve, intuitivamente, ser capaz de reconhecer um número infinito desses pares, caso contrário ele teria que fazer isso deterministicamente a partir de um certo ponto. Ocorrências infinitas desses pares, entretanto, são reconhecidas trivialmente pelo ABT $\mathcal{A}_2$ da Figura 3.1: $L(\mathcal{A}_2) = \{(a^\omega, \overline{\tau}) \mid \forall k \exists i \exists j[(k < i < j) \wedge (\tau_j = \tau_i + 1)]\}$.

Teorema 2 $L(\mathcal{A}_2) \notin \mathcal{ABTQD}$.

Prova. A demonstração é por contradição. Assuma que $\mathcal{B} = \langle \Sigma, Q, Q_0, X, T, F \rangle$ é um ABTQD e que $L(\mathcal{B}) = L(\mathcal{A}_2)$. Nós primeiro escolhemos um palavra temporizada especial $\rho^2 \in L(\mathcal{A}_2)$ e consideramos uma trajetória qualquer de aceitação $r^2 = (\overline{q^2}, \overline{\nu^2})$ de $\mathcal{B}$ sobre ρ^2; depois nós perturbamos ρ^2 de acordo com r^2, obtendo ρ^3, e mostramos que $\mathcal{B}$ possui uma trajetória $r^3 = (\overline{q^3}, \overline{\nu^3})$ sobre ρ^3, tal que $\overline{q^3} = \overline{q^2}$. A contradição é estabelecida quando notamos que $\rho_3 \notin L(\mathcal{A}_2)$.

Seja $n = |\text{Reach}(F)|$ e $k = |X|$. Seja C_B uma constante natural tal que $C_B > 1$ e $C_B > c$ para toda constante $c \in \text{Const}(T)$. Seja $\varepsilon < 1$ uma constante racional tal que para todo $c \in \text{Const}(T) \cup \{1\}$ existe um natural m, tal que $c = m\varepsilon$.

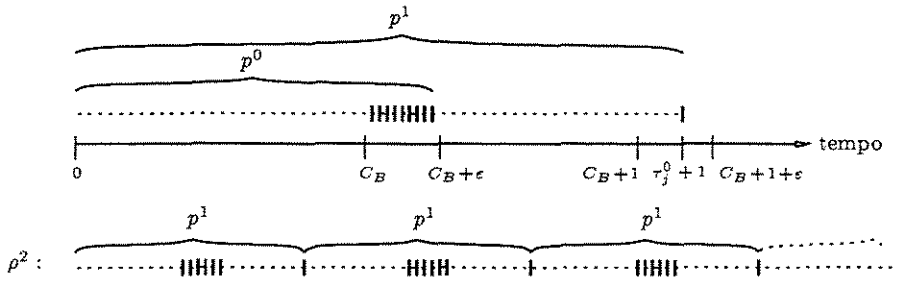

Figura 3.2: Construindo a palavra ρ^2

Para construir ρ^2 nós definimos duas palavras temporizadas finitas. Seja p^0 a seqüência de $(nk+1)$ a's igualmente espaçados entre C_B e $C_B+\varepsilon$, ou seja, $p^0 = (a^{nk+1}, \overline{\tau^0})_{nk+1}$, onde $\tau_i^0 = C_B + \mu i$, para $\mu = \varepsilon/(nk+2)$. A parte superior da Figura 3.2 ilustra p^0. Dado um lugar $\ell \in \text{Reach}(F)$ e uma interpretação de relógio ν, como $\text{Reach}(F)$ é determinístico, existe no máximo uma trajetória finita $(\overline{q}, \overline{\nu})_{nk+1}$ de $\mathcal{B}$ sobre p^0 tal que $(q_0, \nu_0) = (\ell, \nu)$; e, como há k relógios, no mínimo $((n-1)k+1)$ transições nesta trajetória são tais que *nenhum* relógio é zerado *pela última vez* nelas. Além disso, como o valor de qualquer relógio é maior que C_B quando o primeiro a ocorre, exatamente a mesma seqüência de transições será tomada para qualquer ν, quando ℓ é fixo. Assim, existe um índice fixo j, $1 \leq j \leq (nk+1)$, tal que para todo $\ell \in \text{Reach}(F)$ a para todo ν, nenhum relógio é zerado pela última vez na j-ésima transição da trajetória sobre p^0, começando em (ℓ, ν).

Agora, seja $p^1 = (a^{nk+2}, \overline{\tau^1})_{nk+2}$ onde $\tau_i^1 = \tau_i^0$ para $1 \leq i \leq (nk+1)$, e $\tau_{nk+2}^1 = \tau_j^0 + 1$. Então, $\rho^2 = (a^\omega, \overline{\tau^2})$ é a concatenação infinita de p^1, como ilustrado na Figura 3.2. Formalmente, para todo $i > 0$, sejam i^{d} e $i^{\text{m}} < (nk+2)$ dois naturais tais que $i = i^{\text{d}}(nk+2) + i^{\text{m}}$. Com isso, $\tau_i^2 = i^{\text{d}}\tau_{nk+2}^1 + \tau_{i^{\text{m}}}^1$. Note que, quando $i^{\text{m}} \geq 1$, o i-ésimo símbolo de ρ^2 corresponde a um dos $nk+1$ símbolos *iniciais* do prefixo p^1. Quando $i^{\text{m}} = 0$, o i-ésimo símbolo de ρ^2 corresponde ao símbolo final de p^1, que está *casado* (separado por uma distância de exatamente 1 unidade de tempo) do j-ésimo símbolo de p^1. Claramente, $\rho^2 \in L(\mathcal{A}_2)$. Obtemos ρ^3, agora, perturbando ρ^2.

Seja $r^2 = (\overline{q^2}, \overline{\nu^2})$ uma trajetória de aceitação de $\mathcal{B}$ sobre ρ^2. Há ao menos uma tal trajetória porque nós propusemos $L(\mathcal{B}) = L(\mathcal{A}_2)$. Seja f o menor natural tal que $f^{\text{m}} = 0$ e $q_f^2 \in \text{Reach}(F)$. Note que r^2 é determinística da f-ésima transição em diante. Também, para qualquer natural i, se $i \geq f$ e $i^{\text{m}} = 0$, então para todo relógio x, x não é zerado pela última vez na $(i+j)$-ésima transição de r^2. Quer dizer, ou x não é zerado na $(i+j)$-ésima

transição de r^2, ou, se for, então x é zerado na $(i+j')$-ésima transição de r^2, para algum j', $j < j' < nk+2$. Informalmente, esta propriedade faz a trajetória r^2 ser insensível a pequenas perturbações nos tempos de ocorrência τ_i^2, para $i > f$ e $i^{\mathrm{m}} = 0$.

Seja $\rho^3 = (a^\omega, \overline{\tau^3})$ definida fazendo $\tau_i^3 = \tau_i^2 - \mu/2$ se $i > f$ e $i^{\mathrm{m}} = 0$, caso contrário $\tau_i^3 = \tau_i^2$. Assim, $\rho^3 \notin L(\mathcal{A}_2)$. Nós podemos supor, sem perda de generalidade, que $\mathcal{B}$ é completo. Seja $r^3 = (\overline{q^3}, \overline{\nu^3})$ a trajetória de $\mathcal{B}$ sobre ρ^3 tal que $(q_i^3, \nu_i^3) = (q_i^2, \nu_i^2)$ para todo i, $0 \leq i \leq f$. Note que há exatamente uma tal trajetória, pois ρ_3 é igual a ρ_2 até o f-ésimo símbolo, $\mathcal{B}$ é completo e r^3 tem que ser determinística da f-ésima transição em diante.

Nós argumentamos que r^3 e r^2 seguem exatamente a mesma seqüência de transições, o que implica $\overline{q^3} = \overline{q^2}$. A prova é por indução em i. Para $i > f$, se a g-ésima transição de r^3 é igual a g-ésima transição de r^2 para todo $g < i$, então a i-ésima transição de r^3 e r^2 são iguais. Seja $\eta(\mathrm{h})$, $\mathrm{h} \in \{2,3\}$, uma abreviação para $\nu_{i-1}^{\mathrm{h}}(x) + \tau_i^{\mathrm{h}} - \tau_{i-1}^{\mathrm{h}}$. Há dois casos:

1. (símbolos *inicias*) Se $i^{\mathrm{m}} \geq 1$, então para todo $x \in X$, ou $\eta(2) > C_B$ e $\eta(3) > C_B$, ou, $\eta(2) < \varepsilon$ e $\eta(3) < \varepsilon$;

2. (símbolos *casados* em ρ^2) Se $i^{\mathrm{m}} = 0$, então para todo $x \in X$, ou $\eta(2) > C_B$ e $\eta(3) > C_B$, ou para todo natural m: (i) $\eta(2) \leq m\varepsilon$ se e somente se $\eta(3) \leq m\varepsilon$ e; (ii) $\eta(2) \geq m\varepsilon$ se e somente se $\eta(3) \geq m\varepsilon$. Lembre que qualquer constante em $\mathrm{Const}(T)$ é da forma $m\varepsilon$, para algum m, e que qualquer restrição básica é da forma $x \leq c$ ou $x \geq c$, para algum $c \in \mathrm{Const}(T)$.[1]

Os dois casos implicam que, para qualquer restrição de relógio δ em T, $(\nu_{i-1}^3 + \tau_i^3 - \tau_{i-1}^3)$ satisfaz δ se e somente se $(\nu_{i-1}^2 + \tau_i^2 - \tau_{i-1}^2)$ satisfaz δ. Portanto, as i-ésimas transições de r^3 e de r^2 serão idênticas. □

3.2 U_{ABTQD} pertence a Π_1^1

Seja A um ABTQD. Dado um oráculo para uma função f codificando a palavra temporizada racional ρ, se há uma trajetória de aceitação $r = (\overline{q}, \overline{\nu})$ de A sobre ρ, então, *se* é dado um prefixo finito de r, $(\overline{q}, \overline{\nu})_i$, tal que $q_i \in F$, então, como $\mathrm{Reach}(F)$ é determinístico, o

[1] A situação $\eta(2) = \eta(3) = m\varepsilon$ está coberta por este caso mas, na verdade, não pode acontecer, por causa da magnitude da perturbação introduzida em ρ^3.

restante de r é determinado unicamente por ρ; e pode ser construído parametricamente por consultas ao oráculo. Como nós podemos codificar seqüências de estados finitas como *números* naturais, podemos obter um algoritmo Π^1_1 para universalidade, U_{ABTQD}, de ABTQD.

Teorema 3 $U_{\text{ABTQD}} \in \Pi^1_1$.

Prova. Seja $\mathcal{A}_0, \mathcal{A}_1, \ldots$ uma indexação recursiva de todos os ABTQDs. Seja d_5 uma função mapeando números naturais em seqüências de estados finitas. Considere uma MT, M_{H_3}, com um oráculo, que dada a tupla $\langle p, i, j, z\rangle \in \mathbb{N}^4$, tem o seguinte comportamento:

1. Se $j \leq i$, então **rejeita**;
2. Decodifica p de acordo com d_5 obtendo $r = (\overline{q}, \overline{\nu})_k$;
3. Se $k \neq i$, então **rejeita** (que dizer, se p não codifica uma trajetória de tamanho i);
4. Obtém a descrição do autômato $\mathcal{A}_z = \langle \Sigma, Q, Q_0, X, T, F\rangle$, a partir de z;
5. Consulta o oráculo, de acordo com d_1 e d_2 (as funções definidas na página 20), obtendo a palavra temporizada finita $\rho = (\overline{\sigma}, \overline{\tau})_k$;
6. Se r não é uma trajetória de $\mathcal{A}_z$ sobre ρ, ou $q_k \notin F$, então **rejeita**;
7. Consulta o oráculo novamente, de acordo com d_1 e d_2, obtendo a palavra temporizada finita $\rho' = (\overline{\sigma'}, \overline{\tau'})_{k+j}$. Observe que ρ é um prefixo de ρ';
8. Constrói a única trajetória $r' = (\overline{q'}, \overline{\nu'})_{k+j}$ de $\mathcal{A}_z$ sobre ρ', tendo r como prefixo (suponha, sem perda de generalidade, que $\mathcal{A}_z$ é completo);
9. Se $q'_{k+j} \notin F$, então **rejeita**, caso contrário **aceita**.

Temos que $U_{\text{ABTQD}} = \{z \mid \forall f\, [\, (\forall i\, \exists j\, H_1(f,i,j)) \Rightarrow (\exists p\, \forall i\, \exists j\, H_3(f,p,i,j,z))\,]\}$, onde H_1 é a relação definida na página 20 e H_3 a relação cuja MT subjacente é M_{H_3}. □

3.3 U_{ABTQD} é Π_1^1-difícil

A mesma redução da Seção 2.3.2 se aplica a $\mathcal{ABTQD}$, sem dificuldade. A linguagem $\overline{L}$ é aceita pela união disjunta dos seguintes ABTQDs (os nomes vêm de [4]): $\mathcal{A}_0, \mathcal{A}_{\text{init}}, \mathcal{A}_{\text{recur}}$ para as condições de contorno; e $\mathcal{A}_i$, $1 \leq i \leq n$, um autômato para cada instrução de M. Nas descrições seguintes, considere uma palavra temporizada $(\overline{\sigma}, \overline{\tau})$ em $\overline{L}$. Os autômatos são dados pela união disjunta de várias componentes, ilustradas separadamente; por exemplo, $\mathcal{A}_{\text{init}}$ é a união disjunta de $\mathcal{A}^1_{\text{init}}$, $\mathcal{A}^2_{\text{init}}$ e $\mathcal{A}^3_{\text{init}}$.

A Figura 3.3 traz os autômatos para as condições de contorno. Recorde que, na linguagem L, a computação da máquina M é codificada pela seqüência de configurações, $b_{i_1} a_1^{c_1} a_2^{d_1} b_{i_2} a_1^{c_2} a_2^{d_2} \ldots$, onde os b_{i_j} ocorrem nos tempos inteiros, e o número de a_1's e a_2's codificam os valores dos contadores C e D, respectivamente:

- $\mathcal{A}_0^1$ afirma que não há um símbolo b_i no tempo j, para algum $j \geq 1$;
- $\mathcal{A}_0^2$ afirma que o intervalo $(j, j+1)$ não é da forma $a_1^* a_2^*$, para algum $j \geq 1$;
- $\mathcal{A}_{\text{init}}$ afirma que ou $\sigma_1 \neq b_1$ ou $\tau_1 \neq 1$ ou $\tau_2 < 2$;
- $\mathcal{A}_{\text{recur}}$ afirma que $\sigma_j = b_1$ para um número finito de $j \geq 1$.

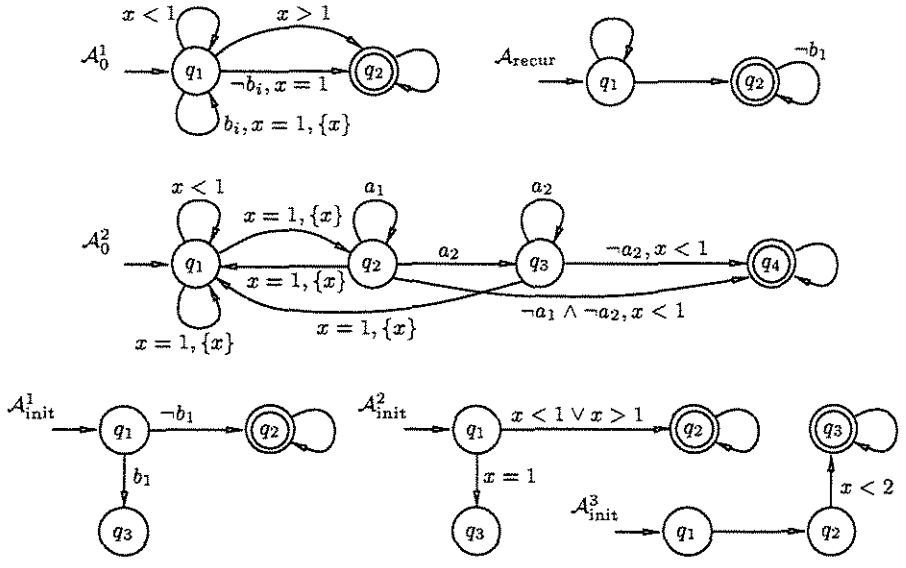

Figura 3.3: Autômatos para as condições de contorno

Dos 6 tipos de instruções, nós consideramos apenas três, pois para os demais usaríamos técnicas semelhantes. Suponha que a instrução 6 é do tipo (d), "incrementar D e pular não-deterministicamente para instrução x ou y", com $x = 11$ e $y = 5$. As Figura 3.4 e 3.5 trazem o autômato$\mathcal{A}_6$:

- $\mathcal{A}_6^1$ afirma que, para algum $j \geq 1$, $\sigma_j = b_6$ não há um b_{11} nem um b_5 no tempo $j+1$;
- Aceitando quando os a_1's não estão casados:
  - $\mathcal{A}_6^2$ afirma que há um b_6 no tempo t, depois um a_1 no tempo $t' \in (t, t+1)$, e não há um a_1 no tempo $t'+1$;
  - $\mathcal{A}_6^3$ e $\mathcal{A}_6^4$ afirmam que há um b_6 no tempo t, um a_1 no tempo $t' \in (t+1, t+2)$, e não ocorre a_1 no tempo $t'-1$;
- Aceitando quando os a_2's não refletem incremento:
  - $\mathcal{A}_6^5$, omitido na Fig. 3.5, é análogo a $\mathcal{A}_6^2$ trocando a_1 por a_2;
  - $\mathcal{A}_6^6$ afirma que há um b_6 no tempo t, e para o último a_2 no tempo $t' \in (t+1, t+2)$ há um a_2 correspondente no tempo $t'-1$;
  - $\mathcal{A}_6^7$ afirma que há b_6 no tempo t, um a_2 no tempo $t' \in (t+1, t+2)$ que *não* é o último em $(t+1, t+2)$, e não há um a_2 no tempo $t'-1$.

Suponha que a instrução 10 é do tipo (b), "decrementar C e pular não-deterministicamente para instrução x ou y", com $x = 9$ e $y = 2$. Para o autômato $\mathcal{A}_{10}$, nós temos:

- $\mathcal{A}_{10}^1$ é análogo a $\mathcal{A}_6^1$ trocando b_6 por b_{10}, b_{11} por b_9, b_5 por b_2;
- Aceitando quando os a_2's não estão casados:
  - $\mathcal{A}_{10}^2$, $\mathcal{A}_{10}^3$ e $\mathcal{A}_{10}^4$ são análogos a $\mathcal{A}_6^2$, $\mathcal{A}_6^3$ e $\mathcal{A}_6^4$, respectivamente, trocando b_6 por b_{10}, a_1 por a_2;
- Aceitando quando os a_1's não refletem decremento:
  - $\mathcal{A}_{10}^5$ e $\mathcal{A}_{10}^6$ são análogos a $\mathcal{A}_6^3$ e $\mathcal{A}_6^4$ trocando b_6 por b_{10};
  - $\mathcal{A}_{10}^7$, na Figura 3.6, afirma que há um b_{10} no tempo t, e o último a_1 no tempo $t' \in (t, t+1)$ tem um a_1 correspondente no tempo $t'+1$;

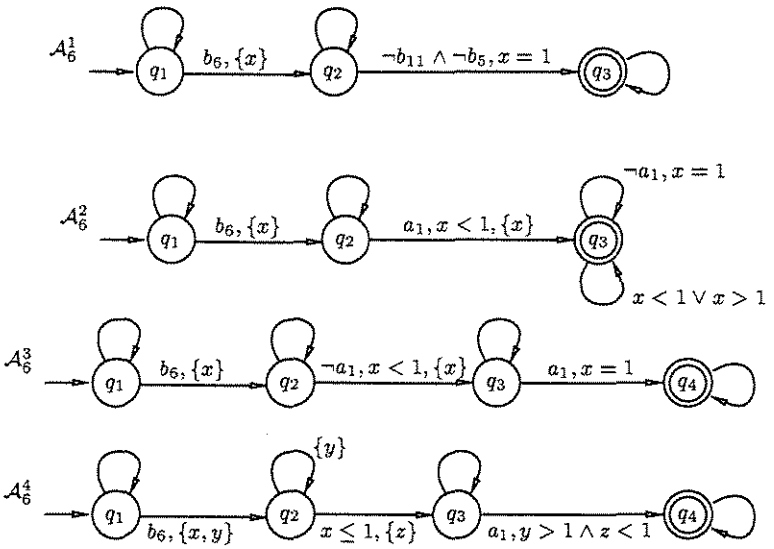

Figura 3.4: Aceitando quando os a_1's não estão casados

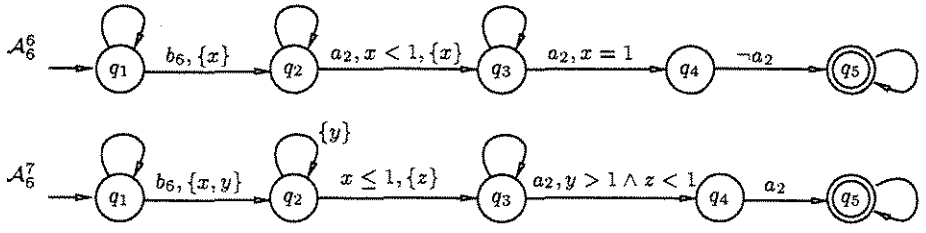

Figura 3.5: Aceitando quando os a_2's não refletem incremento

- $\mathcal{A}_{10}^8$, na Figura 3.6, afirma que há um b_{10} no tempo t, um a_1 no tempo $t' \in (t, t+1)$ que não é o último em $(t, t+1)$, e não há um a_1 no tempo $t'+1$.

Suponha que a instrução 31 é do tipo (f), "se $D = 0$ pular para a instrução x, caso contrário pular para instrução y", com $x = 8$ e $y = 22$. Para o autômato $\mathcal{A}_{31}$, nós temos:

- Aceitando quando a_1's ou a_2's não estão casados:
  - Os autômatos $\mathcal{A}_{31}^1, \mathcal{A}_{31}^2, \ldots, \mathcal{A}_{31}^6$, são análogos, respectivamente, a $\mathcal{A}_6^2$, $\mathcal{A}_6^3$, $\mathcal{A}_6^4$, $\mathcal{A}_{10}^2$, $\mathcal{A}_{10}^3$ e $\mathcal{A}_{10}^4$ trocando b_{31} por b_6, b_{31} por b_{10};
- $\mathcal{A}_{31}^7$, na Figura 3.7, afirma que há um b_{31} no tempo t, depois *nenhum* a_2 em $(t, t+1)$, e não há um b_8 no tempo $t+1$;

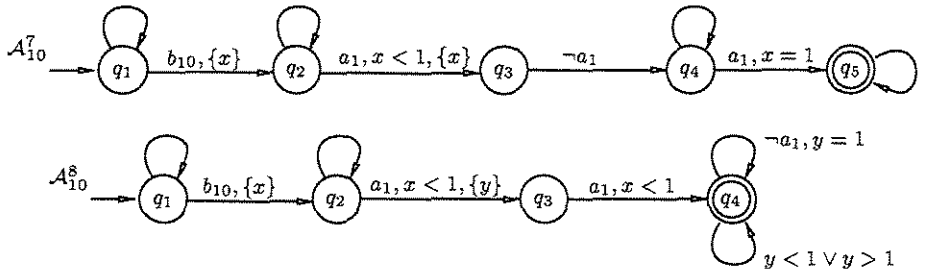

Figura 3.6: Aceitando quando os a_1's não refletem decremento

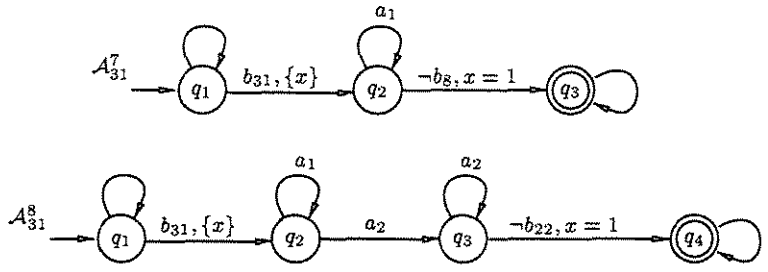

Figura 3.7: Aceitando quando o desvio está *incorreto*

- $\mathcal{A}^8_{31}$, na Figura 3.7, afirma que há um b_{31} no tempo t, depois pelo menos um a_2 em $(t, t+1)$, e não há um b_{22} no tempo $t+1$.

3.4 Variando a Condição de Aceitação

No artigo de Alur e Dill [4], somente as condições de aceitação do tipo Büchi (B) e Muller (M) foram estudadas. Esses dois tipos são geralmente considerados o menos expressivo e o mais expressivo, respectivamente. Num artigo anterior dos mesmos autores [3], eles sugerem a investigação dos outros dois tipos mais comuns, Rabin (R) e Streett (S), notando que eles poderiam definir classes intermediárias de linguagens temporizadas para o caso determinístico. Nesta seção nós propomos definições para autômatos temporizados quase determinísticos do tipo Rabin e Streett, mostrando que esses tipos de condição de aceitação não definem novas classes além daquelas já definidas pelos tipos Büchi e Muller. As definições propostas seguem o espírito das definições correspondentes para ω-autômatos.

Um *autômato (Muller|Rabin|Streett|Büchi) temporizado* $\mathcal{A}$ é uma tabela temporizada

$\langle \Sigma, Q, Q_0, X, T \rangle$ junto com uma condição de aceitação, cuja sintaxe e semântica[2] são dadas pela Tabela 3.1.

	sintaxe	semântica	quase determinístico se
M	$F \subseteq 2^Q$	$inf(r) \in F$	$\forall C \in F\,[\,d(C)\,]$
R	$\{(L_i, U_i)\}_i$ $L_i, U_i \subseteq Q$	$\exists i\,[\,inf(r) \cap L_i \neq \emptyset \wedge$ $inf(r) \cap U_i = \emptyset\,]$	$\forall i\,[\,d(\text{Reach}(L_i)) \vee$ $d(\text{Reach}(\overline{U_i}))\,]$
S	$\{(L_i, U_i)\}_i$ $L_i, U_i \subseteq Q$	$\forall i\,[\,inf(r) \cap L_i = \emptyset \vee$ $inf(r) \cap U_i \neq \emptyset\,]$	$\exists i\,[\,d(\text{Reach}(\overline{L_i})) \wedge$ $d(\text{Reach}(U_i))\,]$
B	$F \subseteq Q$	$inf(r) \cap F \neq \emptyset$	$d(\text{Reach}(F))$

Tabela 3.1: Autômatos Temporizados Quase Determinísticos

Um condição do tipo Büchi pode ser vista como um caso especial tanto de uma do tipo Rabin quanto de uma do tipo Streett, tomando simplesmente os conjuntos de pares $\{(F, \emptyset)\}$ e $\{(Q, F)\}$, respectivamente. As condições Büchi, Rabin e Streett podem ser transformadas numa condição Muller simplesmente tomando exatamente os subconjuntos que satisfazem a condição dada. Por exemplo, para passar da condição Rabin para Muller fazemos $F = \{C \in Q \mid \exists i\,(C \cap L_i \neq \emptyset \wedge C \cap U_i = \emptyset)\}$. Essas observações mostram que $\mathcal{AMT} = \mathcal{ART} = \mathcal{AST} = \mathcal{ABT}$, pois $\mathcal{AMT} = \mathcal{ABT}$ [4].

A última coluna da Tabela 3.1 traz as definições para AMTQD, ARTQD, ASTQD, onde $d(S)$ significa que o conjunto $S \subseteq Q$ é determinístico. Essas restrições puramente sintáticas forçam a Propriedade 1, na página 33.

Na teoria de ω-autômatos, a classe do tipo Büchi determinístico é uma subclasse própria da classe do tipo Muller determinístico. Portanto, em qualquer transformação de um ω-autômato determinístico do tipo Muller num ω-autômato do tipo Büchi equivalente, será "introduzido não-determinismo", como na transformação tradicional dada em [4, pág. 199]. Na verdade, essa transformação tradicional introduz *quase* determinismo, como pode ser visto na generalização dada pelo teorema a seguir.

Teorema 4 $\mathcal{AMTD} \subseteq \mathcal{ABTQD}$.

[2] A notação para a sintaxe e semântica nesta tabela vem de [47].

Prova. Seja $A = \langle \Sigma, Q, Q_0, X, T, F \rangle$ um AMTD. Como $\mathcal{ABTQD}$ é fechada por união, nós podemos supor que $|F| = 1$. Nós construímos um ABTQD equivalente $A' = \langle \Sigma, Q', Q'_0, X', T', F \rangle$. Seja $C = \{c_1, c_2, \ldots, c_k\}$ o único conjunto em F. Então, A' consiste de $k+1$ cópias de A numeradas de 0 a k: $Q' = Q \times \{0, 1, \ldots, k\}$, $Q'_0 = \{(q_0, 0)\}$, onde q_0 é o único lugar em Q_0, e $X' = X$. Informalmente, como ilustrado na Figura 3.8 para um exemplo em que $k = 3$, a cópia 0 é idêntica à A, com a adição de transições não-determinísticas, com origem em C, para a cópia 1. Depois de transitar não-deterministicamente para a cópia 1, uma trajetória é obrigada a circular deterministicamente pelas cópias restantes, sem retornar à cópia 0. Nas cópias 1 a k, há somente transições com origem em C, sendo que uma transição com origem em c_i passa para a cópia $i+1$ (em c_k volta à cópia 1). Isso garante a satisfação da condição Muller, fazendo a condição Büchi ser $F = \{(c_k, k)\}$.

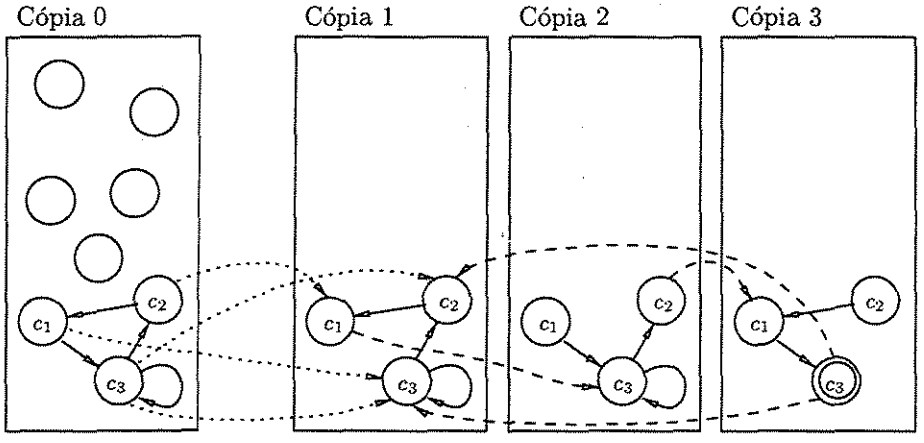

Figura 3.8: Construção para transformar Muller determinístico em Büchi quase determinístico

Formalmente, o conjunto de transições é definido como segue: $\langle q, q', a, \delta, \lambda \rangle \in T'$ se e somente se há $\langle s, s', a, \delta, \lambda \rangle \in T$, $q = (s, i)$, $q' = (s', j)$, e:

1. (dentro da cópia 0) $i = j = 0$; ou
2. (da cópia 0 para a 1) $s \in C$, $s' \in C$, $i = 0$ e $j = 1$; ou
3. (dentro de cada cópia maior que 0) $s \in C$, $s' \in C$, $s \neq c_i$ e $i = j$; ou

4. (da cópia k para a 1) $s = c_i$, $s' \in C$, $i = k$ e $j = 1$; ou

5. (da cópia i para a $i+1$) $s = c_i$, $s' \in C$, $1 \leq i \leq k-1$ e $j = i+1$. □

Corolário 1 $\mathcal{AMTQD} = \mathcal{ARTQD} = \mathcal{ASTQD} = \mathcal{ABTQD}$.

Prova. Usando uma construção similar à do Teorema 4 para mostrar que $\mathcal{AMTQD} \subseteq \mathcal{ABTQD}$; e notando que as condições Büchi, Rabin e Streett podem ser transformadas em condições Muller preservando o quase determinismo. □

Corolário 2 $\mathcal{AMTD} \subset \mathcal{ABTQD}$.

Prova. $\mathcal{AMTD} \subseteq \mathcal{ABTQD}$ pelo Teorema 4, mas $\mathcal{AMTD}$ é fechado por complementação [4] e $\mathcal{ABTQD}$ não é. A inclusão própria segue daí. □

classe	união	interseção	compl.	não-vacuidade	universalidade
$\mathcal{AMT}$	fechado	fechado	aberto	PSPACE-co.	$\mathbf{\Pi_2^1}$, Π_1^1-difícil
$\mathcal{ART}$	fechado	fechado	aberto	PSPACE-co.	$\mathbf{\Pi_2^1}$, Π_1^1-difícil
$\mathcal{AST}$	fechado	fechado	aberto	PSPACE-co.	$\mathbf{\Pi_2^1}$, Π_1^1-difícil
$\mathcal{ABT}$	fechado	fechado	aberto	PSPACE-co.	$\mathbf{\Pi_2^1}$, Π_1^1-difícil
∪					
$\mathcal{AMTQD}$	fechado	fechado	aberto	PSPACE-co.	$\mathbf{\Pi_1^1}$**-completo**
$\mathcal{ARTQD}$	fechado	fechado	aberto	PSPACE-co.	$\mathbf{\Pi_1^1}$**-completo**
$\mathcal{ASTQD}$	fechado	fechado	aberto	PSPACE-co.	$\mathbf{\Pi_1^1}$**-completo**
$\mathcal{ABTQD}$	fechado	fechado	aberto	PSPACE-co.	$\mathbf{\Pi_1^1}$**-completo**
∪					
$\mathcal{AMTD}$	fechado	fechado	fechado	PSPACE-co.	PSPACE-co.
$\mathcal{ARTD}$	fechado	fechado	fechado	PSPACE-co.	PSPACE-co.
$\mathcal{ASTD}$	fechado	fechado	fechado	PSPACE-co.	PSPACE-co.
∪					
$\mathcal{ABTD}$	fechado	fechado	aberto	PSPACE-co.	PSPACE-co.

Tabela 3.2: Um sumário de resultados na teoria de ATQD

Resta considerar autômatos determinísticos. No que se segue, nós suporemos autômatos completos. $\mathcal{ABTD}$ é uma subclasse própria de $\mathcal{AMTD}$ [4] e seria interessante verificar se $\mathcal{ARTD}$ e $\mathcal{ASTD}$ ficam propriamente entre elas. O próximo teorema mostra que este não é o caso.

Teorema 5 $\mathcal{AMTD} = \mathcal{ARTD} = \mathcal{ASTD}$.

Prova. Primeiro notamos que as condições Rabin e Streett são complementares para autômatos determinísticos. Assim, $\mathcal{ARTD} = \overline{\mathcal{ASTD}}$. Como $\mathcal{AMTD}$ é fechada por complementação, é suficiente mostrar que dado um AMTD $A = \langle \Sigma, Q, Q_0, X, T, F \rangle$, nós podemos obter um ARTD $A' = \langle \Sigma, Q', Q'_0, X', T', P \rangle$ equivalente. Como pode ser mostrado que $\mathcal{ARTD}$ é fechado por união, nós podemos assumir que $|F| = 1$. Seja $C = \{c_0, c_1, \ldots, c_{k-1}\}$ o único conjunto em F. Então, A' consiste de k cópias de A, como segue. $Q' = Q \times \{0, 1, \ldots, k-1\}$, $Q'_0 = \{(q_0, 0)\}$, onde q_0 é o único lugar em Q_0 e $X' = X$. O conjunto de transições é definido de modo que qualquer trajetória passa da cópia i para a cópia $(i+1) \pmod k$ sempre que alcança um lugar c_i. Nós definimos $\langle q, q', a, \delta, \lambda \rangle$ em T' se e somente se há $\langle s, s', a, \delta, \lambda \rangle \in T$, $q = (s, i)$, $q' = (s', j)$, e:

1. (dentro de cada cópia) $i = j$ e $s \neq c_i$; ou
2. (da cópia i para a cópia $(i+1) \pmod k$) $j = (i+1) \pmod k$ e $s = c_i$.

Seja $D = \{(c_i, i) \mid 0 \leq i \leq k-1\}$. A condição Rabin é $P = \{(\{(c_0, 0)\}, Q' \setminus D)\}$. □

A Tabela 3.2 sumariza alguns dos resultados na teoria de AT; o que aparece em negrito é resultado das discussões e teoremas desta tese. O símbolo $\bigcup$ significa inclusão própria. É interessante notar que numa tabela análoga para ω-autômatos, todas as classes iriam colapsar em $\mathcal{AMD}$, com exceção de $\mathcal{ABD}$.

3.5 Autômatos de Trajetórias Contáveis

No início deste capítulo dissemos que uma restrição sintática simples tem a vantagem de trazer automaticamente uma indexação recursiva para a classe definida. A questão aqui é que no exercício de encontrar cotas superiores nas hierarquias, às vezes consideramos definições de classes que não são *efetivas* no sentido de implicar numa indexação recursiva. A indexação recursiva parece ser um problema menor nesse exercício. Porém, pode ser um problema sutil e merece discussão.

O objetivo desta seção é apresentar um exemplo extremo de subclasse de $\mathcal{ABT}$, matematicamente bem definida, mas que não é, pelo menos não trivialmente, efetivamente definida de forma a implicar uma indexação recursiva. O interesse nessa classe vem do fato de que é possível mostrar, sem dificuldade, que:

- Ela é, como $\mathcal{ABT}$, fechada por união e interseção;

- É uma superclasse de $\mathcal{ABTQD}$, embora não tenhamos conseguido mostrar se é igual a $\mathcal{ABTQD}$ ou a $\mathcal{ABT}$, ou se está propriamente contida entre essas duas classes;

- *Se* ela possui uma indexação recursiva, então há um algoritmo Σ_2^1 para o problema da universalidade. Note que uma subclasse de $\mathcal{ABT}$, com algoritmo Σ_2^1, também seria um resultado muito interessante para o argumento desta tese.

Notamos primeiro que, intuitivamente, o não-determinismo de um ABT é de grau finito e fixo, quer dizer, em qualquer trajetória sobre qualquer palavra, na ocorrência de um símbolo, o autômato pode ter, digamos, de 1 até no máximo k opções de transições a tomar. Considerando uma ordenação fixa das transições, podemos caracterizar completamente uma dada trajetória pelo lugar inicial, e pela seqüência de opções que o autômato escolhe. Chamamos essa seqüência de opções de uma *guia* para a trajetória.

Dado um ABT $A = \langle \Sigma, Q, Q_0, X, T, F \rangle$, seja k o número máximo de opções não-determinísticas que A pode ter; e seja $\Delta = \{1, 2, \ldots, k\}$. Chamamos um conjunto $C \subseteq \Delta^\omega$ de *completo* para A se, para toda $\rho \in L(A)$, existe $\xi \in \Delta^\omega$, tal que ξ é uma guia para uma trajetória de aceitação de A sobre ρ. O autômato A é *de trajetórias contáveis* (ABTTC) se existe um $C \subseteq \Delta^\omega$, C é completo para A e C é *contável*. Por exemplo, todo ABTD é um ABTTC, fazendo $C = \{1^\omega\}$; há uma só guia para qualquer trajetória sobre qualquer palavra. Todo ABTQD também é um ABTTC, pois toda palavra aceita pelo autômato possui uma guia da forma $\alpha 1^\omega$, onde $\alpha \in \Delta^*$. A questão é: dado um ABT, como saber *efetivamente* se ele é um ABTTC?

Note que um tal conjunto C é um conjunto contável de conjuntos contáveis, portanto pode ser codificado em uma única função natural. Desse modo, *se* a classe $\mathcal{ABTTC}$ possui uma indexação recursiva, então possui um algoritmo Σ_2^1 para universalidade, induzido claramente pela definição: $U_{\text{ADTTC}} = \{z \mid \exists f \forall g [\, (\forall i \exists j H_1(g, i, j)) \Rightarrow (\exists n \exists q \forall i \exists j H(f, g, n, q, i, j, z))\,]\}$, onde a função f codifica um tal conjunto C, g codifica as palavras temporizadas, n e q codificam, respectivamente, qual guia de C usar para g, e em qual lugar M_H deve iniciar a simulação da trajetória.

3.5.1 Não-determinismo Finito

Voltando à classe $\mathcal{ABTQD}$, como vimos, o algoritmo Π_1^1 para sua universalidade foi possível porque os autômatos têm que fazer um número finito de transições não-determinísticas ao aceitar uma palavra. A linguagem $L(\mathcal{A}_2)$ exige, pelo Teorema 2, um número infinito de transições não-determinísticas. Caberia a pergunta: o uso finito do não-determinismo é necessário para um algoritmo Π_1^1? Ou será possível definir uma classe que inclua $L(\mathcal{A}_2)$ e tenha universalidade Π_1^1-completo? O próximo capítulo define e discute, justamente, uma classe que responde afirmativamente a segunda pergunta.

Capítulo 4

Autômatos Marcadores de Passo

A propriedade essencial que define autômatos marcadores de passo, que se aplica genericamente a qualquer tipo de ω-autômato, está ilustrada na Figura 4.1:

Propriedade 2 (1) Se q é um lugar final que aparece infinitas vezes em alguma trajetória do autômato sobre uma dada palavra ρ, então (2) qualquer trajetória finita do autômato, sobre um prefixo finito de ρ, que termine no lugar q, é prefixo de alguma trajetória (3) do autômato sobre ρ, na qual q aparece infinitas vezes.

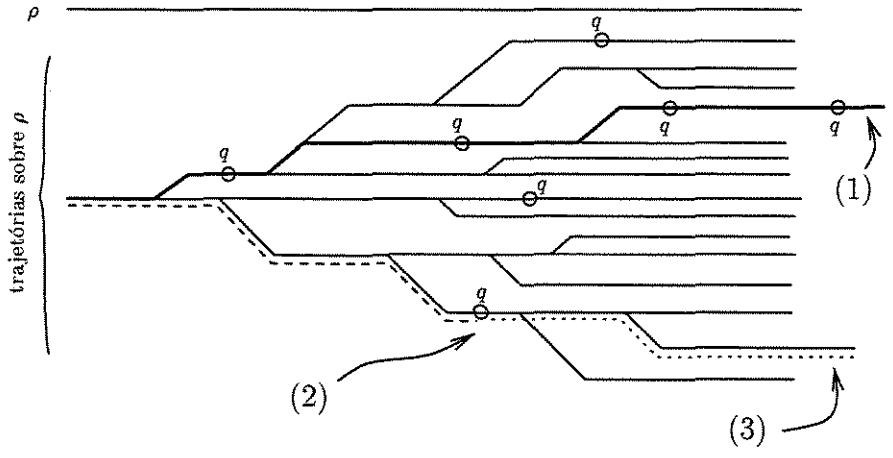

Figura 4.1: Ilustração da propriedade para marcação de passo

Essa propriedade, que permitirá a obtenção de um algoritmo Π^1_1 para universalidade, pode ser forçada por uma restrição sintática, como ilustrado na Figura 4.2: toda transição

com origem num lugar final não possui restrição sobre os relógios, zera todos os relógios, e tem como destino um lugar onde há pelo menos uma transição, com as mesmas características, com destino no mesmo lugar.

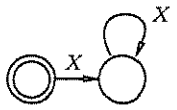

Figura 4.2: Restrição sintática para autômatos marcadores de passo

Um ABT $A = \langle \Sigma, Q, Q_0, X, T, F \rangle$ é *marcador de passo* (ABTMP) se para toda $\langle q, q', a, \delta, \lambda \rangle \in T$, se $q \in F$, então $\delta = \texttt{true}$, $\lambda = X$ e para todo $b \in \Sigma$, $\langle q, q', b, \texttt{true}, X \rangle \in T$ e $\langle q', q', b, \texttt{true}, X \rangle \in T$. Dois exemplos de ABTMP são os próprios $\mathcal{A}_1$ e $\mathcal{A}_2$ da Figura 3.1.

Lemas de Iteração. Uma trajetória de um autômato marcador de passo pode fazer um número infinito de transições não-determinísticas. Por isso, o autômato pode verificar, como $\mathcal{A}_2$ faz, que uma dada palavra possui infinitos pares de símbolos separados por um intervalo unitário. Intuitivamente, porém, o autômato não tem "controle" sobre o que ocorre entre cada um dos pares que ele identifica. Isso é o que, em essência, permitirá um algoritmo Π^1_1 para universalidade.

Sejam $A = \langle \Sigma, Q, Q_0, X, T, F \rangle$ um ABTMP e $\rho \in L(A)$. Os lemas seguintes podem ser provados sem dificuldade:

Lema 2 *Seja $r = (\overline{q}, \overline{\nu})$ uma trajetória de aceitação de A sobre ρ. Seja $q \in F \cap \inf(r)$* e quaisquer *$i < j$, tal que $q_i = q_j = q$. Então a seguinte trajetória r' de A também é de aceitação sobre ρ, com $q \in \inf(r')$: $r' = (\overline{q'}, \overline{\nu'})$, onde $q'_k = q_{j+1}$ e $\nu'_k = \nu_{j+1}$ se $i + 1 \leq k \leq j + 1$; caso contrário $q'_k = q_k$ e $\nu'_k = \nu_k$.*

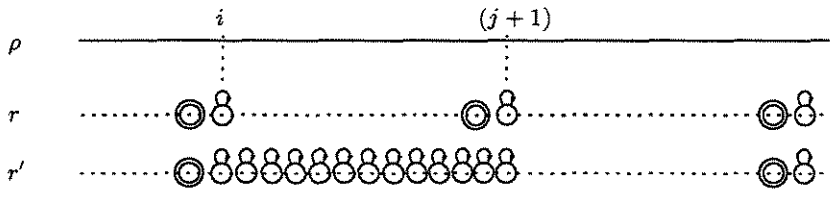

Figura 4.3: Autômato marcando passo

Quer dizer, conforme ilustra a Figura 4.3, a trajetória r' é igual a r até a i-ésima transição, depois pula para q_{j+1} e fica nesse lugar "marcando passo" até o $(j+1)$-ésimo símbolo de ρ, e depois segue igual a r. Esse lema dá, potencialmente, novas trajetórias de aceitação para a mesma palavra ρ, a partir da trajetória r. Os próximos lemas, que serão usados mais a frente para demonstrar que certas linguagens não pertencem a $\mathcal{ABTMP}$, dão novas palavras em $L(A)$ a partir da trajetória r. Note que, quando o autômato está marcando passo, ele é, na verdade, insensível aos símbolos e seus tempos de ocorrência. Dessa forma, podemos alterar a palavra ρ, preservando a pertinência em $L(A)$.

Lema 3 *Seja k tal que existe uma trajetória $r = (\overline{q}, \overline{\nu})$ de A sobre ρ, tal que $q_k \in F \cap \inf(r)$. Então para todo $\gamma \in \Sigma^t$, se $\gamma_{[1,k]} = \rho_{[1,k]}$, então para todo $i > k$, existe $j > i$ tal que $\gamma_{[1,j]} \cdot \rho_{[j+1)} \in L(A)$.*

De forma mais geral, temos o Lema da Iteração, onde Σ^{ft} denota o conjunto de todas as palavras temporizadas finitas:

Lema 4 (Lema da Iteração) *Se $L \in \mathcal{ABTMP}$, então para toda $\rho \in L$, existe k e l, $k < l$, tal que para qualquer $\gamma \in \Sigma^{ft}$, $\rho_k \cdot \gamma \cdot \rho_{[l+1)} \in L$.*

4.1 Propriedades de Fechamento

A classe $\mathcal{ABTMP}$ é fechada por união, mas não por complementação. Os mesmos argumentos para $\mathcal{ABTQD}$, no início do Capítulo 3, se aplicam neste caso. Para fechamento por interseção, entretanto, é necessária uma versão modificada da construção do produto, que na sua forma tradicional não gera necessariamente um ABTMP. O teorema seguinte descreve essa nova construção e é um exemplo interessante de como a restrição de marcação de passo reduz a complexidade do conjunto de trajetórias de um autômato sobre uma dada palavra temporizada.

Teorema 6 *$\mathcal{ABTMP}$ é fechada por interseção.*

Prova. Sejam $A_1 = \langle \Sigma, Q_1, Q_{01}, X_1, T_1, F_1 \rangle$ e $A_2 = \langle \Sigma, Q_2, Q_{02}, X_2, T_2, F_2 \rangle$ dois dados ABTMPs. Assumimos, sem perda de generalidade, pois $\mathcal{ABTMP}$ é fechada por união, que $|F_1| = |F_2| = 1$. Sejam $p \in F_1$ e $s \in F_2$. Dada $\rho \in L(A_1) \cap L(A_2)$, tome quaisquer duas trajetórias de aceitação $r^1 = (\overline{q^1}, \overline{\nu^1})$ e $r^2 = (\overline{q^2}, \overline{\nu^2})$, de A_1 e A_2, respectivamente,

sobre ρ. Então, haverá dois lugares $p' \in Q_1$ e $s' \in Q_2$, tal que p e p' aparecem consecutivamente infinitas vezes em $\overline{q^1}$, e s e s' aparecem consecutivamente infinitas vezes em $\overline{q^2}$. A construção abaixo dará um ABTMP $A_{p's'}$ que aceita exatamente as palavras para as quais há tais trajetórias com p' e s'. O ABTMP que aceita a interseção $L(A_1) \cap L(A_2)$ pode ser obtido com a união de várias dessas construções, uma para cada possível par de lugares $p' \in Q_1$ e $s' \in Q_2$.

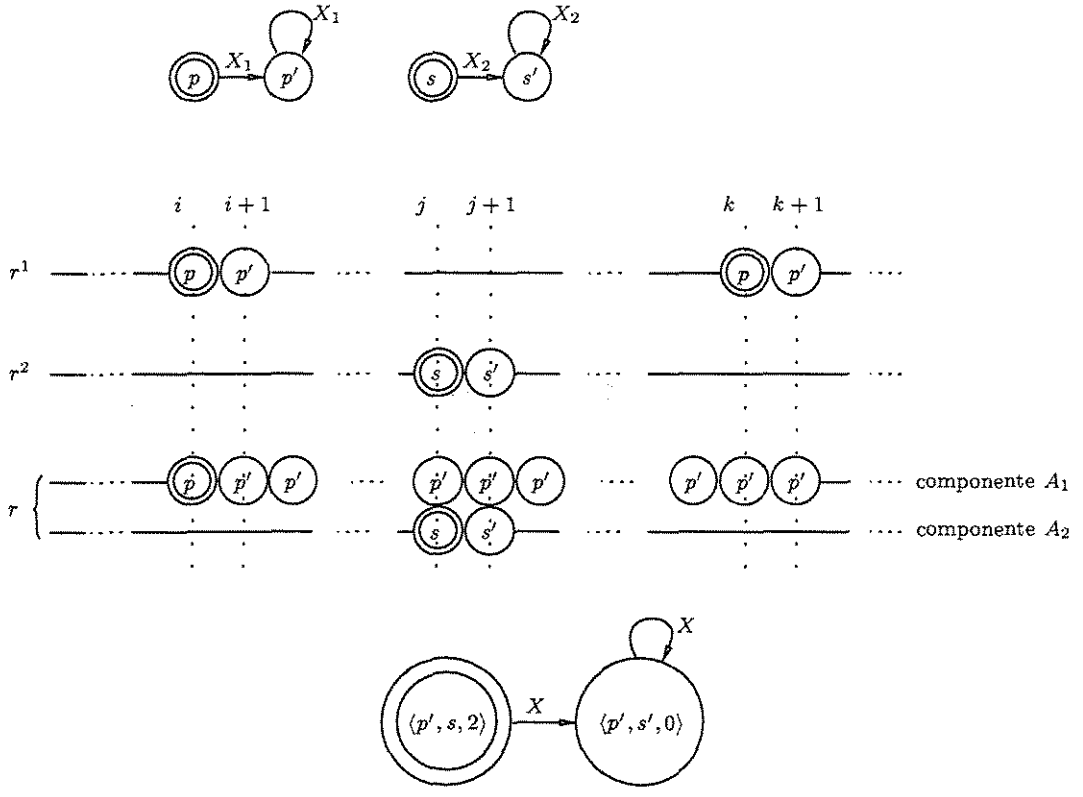

Figura 4.4: Interseção de ABTMP

Agora, como ilustrado na Figura 4.4, seja i um índice tal que $q_i^1 = p$ e $q_{i+1}^1 = p'$. Seja $j > i$ um índice tal que $q_j^2 = s$ e $q_{j+1}^2 = s'$; e $k > j$ tal que $q_k^1 = p$ e $q_{k+1}^1 = p'$. Os índices j e k existem, pois os pares p e p', e s e s', ocorrem infinitas vezes em r^1 e r^2, respectivamente.

Como A_1 pode marcar passo em p' entre quaisquer duas ocorrências do par pp' em r^1, o autômato produto tradicional terá uma trajetória $r = (\overline{q}, \overline{\nu})$ sobre ρ, onde $q_j = \langle p', s \rangle$ e $q_{j+1} = \langle p', s' \rangle$. Nessa trajetória, a componente A_2 da trajetória r está imitando r^2.

A componente A_1 imita r^1 até a posição i, marca passo no lugar p' até a posição k, depois continua seguindo r^1; veja ilustração na Figura 4.4. Repetindo infinitas vezes essa mesma seqüência de argumentos, para novos índices i, j e k a cada vez maiores do que os anteriores, podemos ver que o produto tradicional terá uma outra trajetória r' sobre ρ onde o par $\langle p', s\rangle\langle p', s'\rangle$ ocorre infinitas vezes. Como $\langle p', s'\rangle$ possui, por construção, um "loop" sem restrições, o produto tradicional, embora não sendo um ABTMP, pode, de fato, marcar passo entre quaisquer duas ocorrências do par $\langle p', s\rangle\langle p', s'\rangle$ em r'.

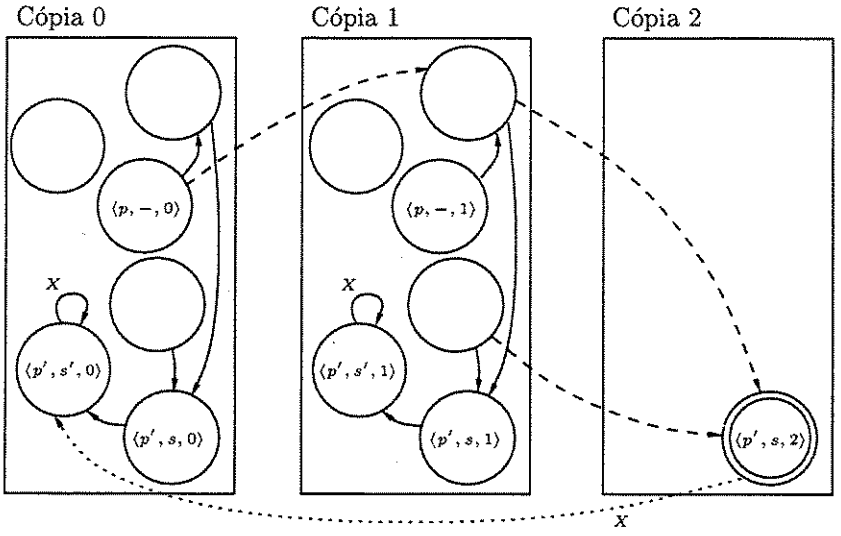

Figura 4.5: Ilustração do autômato produto para interseção

Essa observação leva à construção desejada. Ela é composta por três cópias do produto tradicional, como ilustrado na Figura 4.5. Uma trajetória pode passar da cópia 0 para a cópia 1 após a componente A_1 passar pelo lugar p. Para toda transição, na cópia 1, com destino no lugar $\langle p', s\rangle$, há uma transição equivalente para a cópia 2, onde o lugar $\langle p', s\rangle$ aparece como o único lugar final da construção. $A_{p's'} = \langle \Sigma, Q, Q_0, X, T, F\rangle$, onde:

- $Q = Q_1 \times Q_2 \times \{0, 1, 2\}$;
- $Q_0 = \{\langle a, b, 0\rangle \mid a \in Q_{1_0} \text{ e } b \in Q_{2_0}\}$;
- $X = X_1 \cup X_2$, união disjunta;
- $F = \{\langle p', s, 2\rangle\}$;

- $e = \langle\langle a, b, i\rangle, \langle a', b', i'\rangle, \sigma, \delta, \lambda\rangle \in T$ se e somente se:

  - (da cópia 2 para a cópia 0) $e = \langle\langle p', s, 2\rangle, \langle p', s', 0\rangle, \sigma, \texttt{true}, X\rangle$, $\sigma \in \Sigma$;
  - $\langle a, a', \sigma, \delta_1, \lambda_1\rangle \in T_1$, $\langle b, b', \sigma, \delta_2, \lambda_2\rangle \in T_2$, $\lambda = \lambda_1 \cup \lambda_2$, $\delta = \delta_1 \wedge \delta_2$, e:
    * (dentro da cópia 1) $i = i' = 1$; ou
    * (dentro da cópia 0) $i = i' = 0$; ou
    * (da cópia 0 para a cópia 1) $a = p$, $i = 0$ e $i' = 1$; ou
    * (da cópia 1 para a cópia 2) $i = 1$, $i' = 2$, $a' = p'$ e $b' = s$. □

4.2 Expressividade

Como tradicionalmente em Teoria de Linguagens Formais, o Lema 4, da Iteração, é uma ferramenta poderosa para mostrar que certas linguagens não pertencem à classe $\mathcal{ABTMP}$. A marcação de passo nesse tipo de autômato provoca, intuitivamente, um corte ortogonal na seqüência crescente de expressividade dada por $\mathcal{ABTD}$, $\mathcal{ABTQD}$ e $\mathcal{ABT}$. Linguagens conceitualmente bastante simples não podem ser aceitas por ABTMPs. Um exemplo extremo é $\{(a^\omega, \overline{\tau}) | \forall i,\ \tau_i = i\}$, uma única palavra com a's espaçados regularmente pelo intervalo unitário. O Lema da Iteração pode ser aplicado diretamente nesse caso. Essa linguagem, entretanto, pode ser aceita por um ABTD. Abaixo damos outro exemplo, mais interessante, que está, como será discutido no Capítulo 5, fora da união $\mathcal{ABTQD} \cup \mathcal{ABTMP}$.

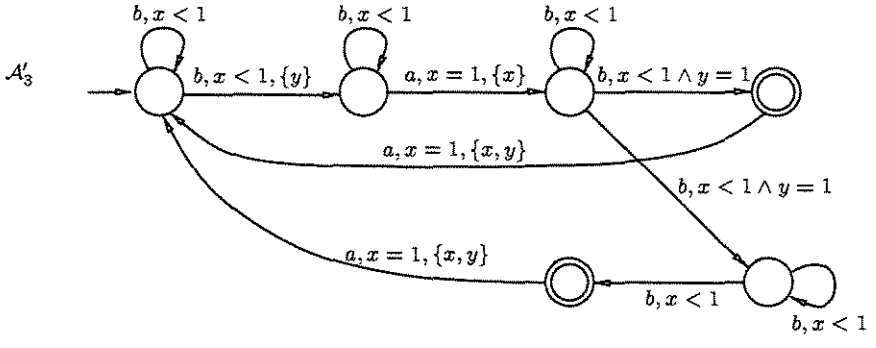

Figura 4.6: $\mathcal{ABTMP}$ é subclasse própria de $\mathcal{ABT}$

Considere o ABT $\mathcal{A}'_3$ da Figura 4.6. Além de a's regularmente espaçados pelo intervalo unitário, ele afirma que em cada par de intervalos consecutivos há um par de b's também espaçados por uma unidade de tempo. Rigorosamente, a linguagem que ele aceita é $L(\mathcal{A}'_3) = \{(\overline{\sigma} \in (b^+a)^\omega, \overline{\tau}) \mid \forall i[\,(\tau_i \in \mathbb{N} \Leftrightarrow \sigma_i = a) \wedge (\exists j,\, \tau_j = i) \wedge (\exists j \exists k,\, 2i < \tau_j < 2i+1,\, \tau_k = \tau_j + 1)\,]\}$. Novamente, uma aplicação direta do Lema da Iteração demonstra o teorema seguinte.

Teorema 7 $L(\mathcal{A}'_3) \notin \mathcal{ABTMP}$.

4.3 U_{ABTMP} é Π_1^1-completo

Considere o seguinte procedimento paramétrico para verificar se um ABT A possui uma trajetória, sobre uma dada palavra temporizada ρ, que passa infinitas vezes por um dado lugar final s: consultando um oráculo que codifica ρ, simule, por busca em largura, todas as possíveis trajetórias finitas de A sobre ρ até alguma trajetória finita r_1 alcançar um estado da forma $\langle s, - \rangle$; depois simule, em largura, todas as possíveis trajetórias finitas de A sobre o restante de ρ, tendo r_1 como prefixo, até alguma trajetória r_2 alcançar um estado da forma $\langle s, - \rangle$; e assim por diante. Se um estado da forma $\langle s, - \rangle$ for alcançado infinitas vezes, então declare $\rho \in L(A)$, caso contrário declare $\rho \notin L(A)$. Claramente, esse procedimento sempre acerta quando declara $\rho \in L(A)$; porém, pode errar quando declara $\rho \notin L(A)$.

A demonstração do teorema seguinte mostra que, para ABTMP, esse procedimento sempre acerta, nos dois casos, e leva a um algoritmo Π_1^1 para universalidade.

Teorema 8 $U_{\text{ABTMP}} \in \Pi_1^1$.

Prova. Seja $\mathcal{A}_0, \mathcal{A}_1, \ldots$ uma indexação recursiva de todos os ABTMPs. Considere uma máquina de Turing M_{H_4} com um oráculo que, para $\langle s, i, j, z \rangle \in \mathbb{N}^4$, tem o seguinte comportamento:

1. Constrói $\mathcal{A}_z = \langle \Sigma, Q, Q_0, X, T, F \rangle$;
2. Se $s \notin F$, então **rejeita**;
3. Consulta o oráculo, de acordo com d_1 e d_2, obtendo a palavra temporizada finita $\rho = (\overline{\sigma}, \overline{\tau})_j$;

4. Faz $S = \{\langle q, \nu_0\rangle \mid q \in Q_0\}$, onde $\nu_0(x) = 0$ para todo $x \in X$;

5. Faz $k = 0$;

6. Para ℓ variando de 1 a j executa:

   - Faz $S' = \{\langle q', \nu'\rangle \mid$ existe $\langle q, \nu\rangle \in S$ e existe $\langle q, q', \sigma_\ell, \delta, \lambda\rangle \in T$ tal que, $(\nu + \tau_\ell - \tau_{\ell-1})$ satisfaz δ, e $\nu'(x) = 0$ se $x \in \lambda$, caso contrário $\nu'(x) = \nu(x) + \tau_\ell - \tau_{\ell-1}$;
   - Se há algum $\langle s, -\rangle \in S'$, então faz $k = k + 1$ e faz $S = \{\langle s, \nu^s\rangle\}$, onde $\langle s, \nu^s\rangle$ é o primeiro $\langle s, -\rangle$ em S', para alguma ordem fixa; caso contrário faz $S = S'$;

7. Se S tem apenas um $\langle s, -\rangle$ e $k = i$, então **aceita**; caso contrário **rejeita**.

Temos que $U_{\text{ABTMP}} = \{z \mid \forall f\,[\,(\forall i\,\exists j\, H_1(f, i, j)) \Rightarrow (\exists s\,\forall i\,\exists j\, H_4(f, s, i, j, z))\,]\}$. Se f codifica uma palavra temporizada, quer dizer, se $\forall i\,\exists j\, H_1(f, i, j)$, e vale $\exists s\,\forall i\,\exists j\, H_4(f, s, i, j, z)$ então claramente, a palavra é aceita por $\mathcal{A}_z$. Para a outra direção, se $\mathcal{A}_z$ tem uma trajetória de aceitação $r = (\overline{q}, \overline{\nu})$ sobre a palavra temporizada ρ codificada por f, então s pode ser qualquer lugar em $inf(r) \cap F$. Seja ℓ o primeiro natural tal que $q_\ell = s$. Então, certamente, vale $H_4(f, s, 1, \ell', z)$, para algum $\ell' \leq \ell$. Para efeito de indução, precisamos mostrar que para $i \geq 1$, se existe j tal que $H_4(f, s, i, j, z)$, então a existência da trajetória r implica que haverá, necessariamente, um j' tal que $H_4(f, s, i + 1, j', z)$.

Seja $r' = (\overline{q'}, \overline{\nu'})_j$ a trajetória seguida por M_{H_4} para a tupla $\langle f, s, i, j, z\rangle$. Considere qualquer m e n, $j < m < n$, tal que $q_m = q_n = s$. Então A_z possui a seguinte trajetória r'' sobre $\rho_{[1,n]}$: $r'' = (\overline{q''}, \overline{\nu''})_n$, onde

- (antes da marcação) $q''_k = q'_k$ e $\nu''_k = \nu'_k$ se $0 \leq k \leq j$;
- (*marcação de passo*) $q''_k = q_{m+1}$ e $\nu''_k = \nu_{m+1}$ se $j + 1 \leq k \leq m + 1$;
- (depois da marcação) $q''_k = q_k$ e $\nu''_k = \nu_k$, caso contrário.

Portanto, há j', $j < j' \leq n$, tal que $H_4(f, s, i + 1, j', z)$. □

A Π^1_1-dificuldade de U_{ABTMP} também é um corolário da Π^1_1-dificuldade de U_{ABT}, pois, assim como no Capítulo 3, todas as linguagens temporizadas necessárias para a redução podem ser aceitas por ABTMPs. Todos os autômatos da Seção 3.3 são, também, ABTMPs ou podem ser transformados facilmente em ABTMPs. A única exceção é o autômato

$\mathcal{A}_{\text{recur}}$, cuja linguagem não pode ser aceita por um ABTMP. Ela afirma que a computação da máquina de Turing de dois contadores *não* é recorrente. Entretanto, esta linguagem é, na verdade, desnecessária na redução. A questão da recorrência era importante no artigo [19], porém o problema de decidir se uma máquina de Turing não-determinística tem uma computação infinita (não necessariamente recorrente), sobre a fita vazia, também pode ser demonstrado Σ_1^1-completo, com a mesma técnica de [19].

4.4 Autômatos Zerados

O Lema da Iteração para ABTMP não tem exatamente a forma tradicional desse tipo de lema; ele é uma espécie de "super" lema da iteração. A palavra temporizada tem segmentos que podem ser substituídos por *qualquer* outro segmento e não apenas por produtos do próprio segmento. Nesta seção apresentamos a definição de uma superclasse própria de $\mathcal{ABTMP}$, autômatos *zerados*, que admite apenas um Lema da Iteração na forma tradicional. Com esse lema podemos mostrar facilmente que essa classe ainda é subclasse própria de $\mathcal{ABT}$. O ponto de interesse aqui é que já não conseguimos obter um algoritmo Π_1^1 para universalidade. É um exemplo de classe, intuitivamente bem menos expressiva do que $\mathcal{ABT}$, mas que ainda apresenta as mesmas dificuldades nas tentativas de obtenção de um algoritmo Π_1^1.

Toda trajetória de um autômato zerado, após passar por um lugar final, tem todos os seus relógios zerados. Quer dizer, o autômato, após passar por um lugar final, perde a memória sobre os tempos de ocorrência de símbolos passados. Um ABT $A = \langle \Sigma, Q, Q_0, X, T, F \rangle$ é *zerado* (ABTZ) se para toda $\langle q, q', a, \delta, \lambda \rangle \in T$, se $q \in F$, então $\lambda = X$. Claramente, todo ABTMP é um ABTZ. Os exemplos anteriores, $\mathcal{A}_0$, $\mathcal{A}_{\text{safra}}$, $\mathcal{A}_1$, $\mathcal{A}_2$ e $\mathcal{A}'_3$, nas páginas 15, 29, 34, 34 e 54, respectivamente, são todos ABTZs.

O Lema da Iteração para ABTZ tem a seguinte forma e pode ser demonstrado sem dificuldade:

Lema 5 (Lema da Iteração) *Se $L \in \mathcal{ABTZ}$, então para toda $\rho \in L$, existe k e l, $k < l$, tal que para qualquer $n \geq 0$, $\rho_k \cdot \rho_{[k+1,l]}^n \cdot \rho_{[l+1)} \in L$.*

Expressividade. Assim como $\mathcal{ABTMP}$, a classe $\mathcal{ABTZ}$ não inclui linguagens que podem ser aceitas trivialmente por ABTDs. Um exemplo é a linguagem aceita pelo autômato

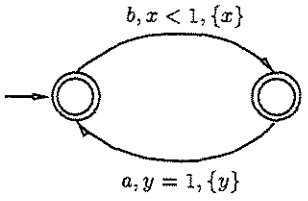

Figura 4.7: Autômato determinístico aceitando linguagem fora de $\mathcal{ATBZ}$

da Figura 4.7. Além de a's regularmente espaçados por intervalos unitários, entre cada intervalo há um b. Mais ainda, a distância entre dois b's consecutivos é sempre menor do que 1. Isso força a distância entre um a e o b seguinte ser estritamente decrescente. Formalmente, a linguagem aceita é $\{((ba)^\omega, \overline{\tau}) \mid \forall i\,[\,(\tau_{2i} = i) \wedge (\tau_i \notin \mathbb{N} \Rightarrow \sigma_i = b) \wedge (\tau_{i+2} - \tau_i < 1)]\}$. Uma aplicação direta do Lema 5 mostra que essa linguagem não pode ser aceita por um ABTZ. Qualquer escolha de k, l e $n > 1$ levará a uma palavra fora da linguagem.

A linguagem $L(\mathcal{A}'_3)$, da Figura 4.6, está fora da união $\mathcal{ABTQD} \cup \mathcal{ABTMP}$. No próximo capítulo apresentaremos a definição de autômatos *finais*, que inclui essa linguagem e foi motivada pelo autômato $\mathcal{A}'_3$. Embora não esteja contida na união $\mathcal{ABTQD} \cup \mathcal{ABTMP}$, a classe de linguagens aceitas por autômatos finais possui universalidade Π^0_1-completo, ou co-recursivamente enumerável completo. Ela é um exemplo de classe cujo grau de indecidibilidade da universalidade é o *menor* possível, entre os completos para algum nível das hierarquias.

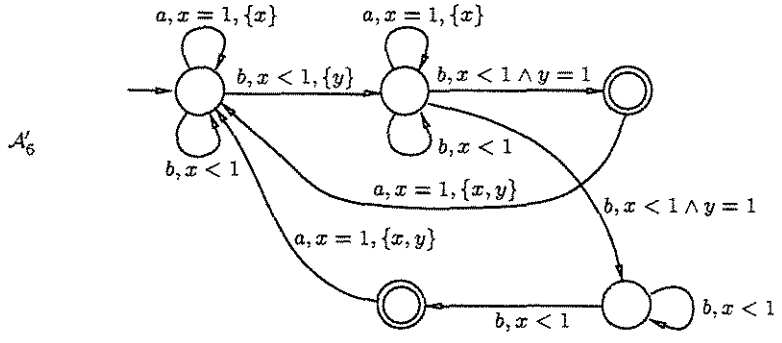

Figura 4.8: L_6 pertence a $\mathcal{ABTZ}$

No Capítulo 5 veremos que a classe definida aqui nesta seção, $\mathcal{ABTZ}$, pode aceitar linguagens fora da classe de autômatos finais e fora da união $\mathcal{ABTQD} \cup \mathcal{ABTMP}$. Um

exemplo é a linguagem do autômato $\mathcal{A}'_6$, apresentado na Figura 4.8, que será definida no Capítulo 5. Para finalizar, notamos também que esta classe $\mathcal{ABTZ}$ é fechada por união, mas não por complementação; e que não conseguimos demonstrar o fechamento por interseção. Note que a técnica utilizada no Teorema 6 é inútil neste caso.

Capítulo 5

Autômatos Finais e o Relacionamento entre as Classes

Note que o autômato $\mathcal{A}'_3$, na Figura 4.6, possui a seguinte característica: nenhuma trajetória pode ficar circulando infinitamente em qualquer dos laços simples do autômato; antes de uma unidade de tempo, a trajetória é forçada a trocar de lugar, ou então terminar finitamente. Com isso, vale para aquele autômato a seguinte propriedade, que também se aplica genericamente a qualquer tipo de ω-autômato:

Propriedade 3 Qualquer trajetória é de aceitação.

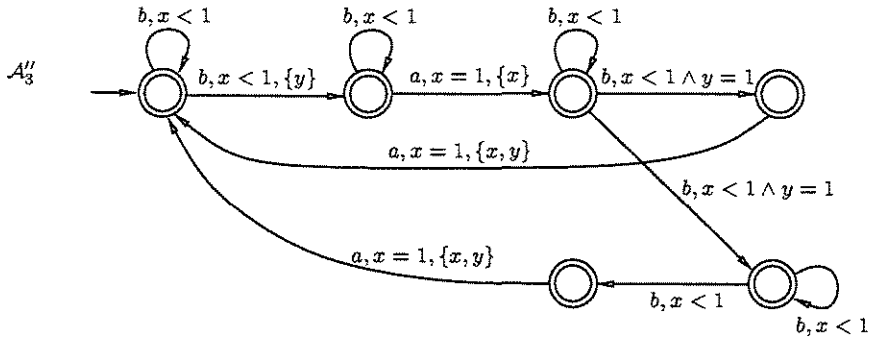

Figura 5.1: Os autômatos $\mathcal{A}''_3$ e $\mathcal{A}'_3$ são equivalentes

Note ainda que, para qualquer autômato que possua essa propriedade, podemos obter um outro autômato equivalente transformando todos os lugares em finais, como exempli-

ficado na Figura 5.1. Chamamos um ABT $A = \langle \Sigma, Q, Q_0, X, T, F \rangle$ de *final* (ABTF) se $F = Q$. A classe $\mathcal{ABTF}$ é, como as outras, fechada por união e por interseção, mas não por complemento. Os argumentos são similares àqueles apresentados no Capítulo 3.

Nesta seção veremos que $\mathcal{ABTF}$ é subclasse própria de $\mathcal{ABT}$ e que U_{ABTF} é Π_1^0-completo. Esses resultados foram obtidos antes que percebêssemos que, na verdade, a interpretação mais precisa para a propriedade 3 é de que a condição de aceitação do autômato é inócua. O autômato aceita uma palavra se existe uma trajetória sobre ela, simplesmente. Esse tipo de autômato temporizado *sem* condição de aceitação é chamado em [25] de *timed safety automata*. Naquele artigo, e também em [29], a indecidibilidade da universalidade, mas não seu grau, já havia sido estabelecida, assim como a expressividade da classe em relação a $\mathcal{ABT}$.

O valor do novo resultado, mostrando que U_{ABTF} é Π_1^0-completo, é que a "alta" indecidibilidade da universalidade para $\mathcal{ABT}$, $\mathcal{ABTQD}$ e $\mathcal{ABTMP}$ não vem unicamente da combinação entre o não-determinismo e a temporização dos autômatos, mas de uma *combinação* desses fatores com as condições de aceitação. A diferença entre um ABTF e um ABT está apenas na condição de aceitação, mas a complexidade da universalidade salta de completa para o primeiro nível da hierarquia *aritmética* para difícil para o primeiro nível da hierarquia *analítica*. Por que ocorre esse grande salto? Uma explicação intuitiva é que a presença de condições de aceitação faz com que o autômato possua um mecanismo para diferenciar dois tipos de trajetórias *infinitas*. Um autômato com $Q = F$, na essência, é capaz apenas de diferenciar entre trajetórias finitas e infinitas. A diferença entre a hierarquia aritmética e a analítica é, justamente, a quantificação (que também pode ser pensada como um mecanismo de diferenciação) sobre funções infinitas.

Os diversos tipos de condições de aceitação, para autômatos do tipo ω, surgiram não apenas como uma maneira de forçar a passagem das trajetórias do autômato por certos conjuntos de lugares, ou como uma forma de expressar propriedades do tipo *safety* e *liveness* [21]; as condições de aceitação também são, assim como o não-determinismo, uma ferramenta que facilita a modelagem de propriedades gerais. Como já foi comentado, e será mostrado em seguida, a classe autômatos temporizados sem condição de aceitação, igual a $\mathcal{ABTF}$, é menos expressiva do que $\mathcal{ABT}$. Os artigos [25, 29] mostram que, para essa classe menos expressiva, todos os tipos de condições de aceitação, Büchi, Muller, Rabin e Streett, podem ser simulados usando a temporização dos autômatos. Quer dizer,

relógios e restrições nas transições podem ser usados para forçar, por exemplo, que todas as trajetórias passem infinitas vezes por determinado lugar. Com isso, esses artigos, e alguns outros artigos da literatura, defendem a tese de condições de aceitação são inúteis para formalismos do tipo ω temporizados não-determinísticos. Porém, para concordar com essa tese, é preciso aceitar a redução na expressividade dos autômatos. Para motivações de aplicação prática, talvez a expressividade de $\mathcal{ABTF}$ seja suficiente, e podemos concordar; mas, do ponto de vista teórico, a tese é apenas mais uma possibilidade interessante. Essa discussão é um exemplo da relação entre expressividade e complexidade de problemas de decisão.

Expressividade. Note que mesmo com o não-determinismo, o conjunto de trajetórias de um autômato temporizado qualquer, sobre uma dada palavra, toma a forma de uma árvore em que cada nó pode ter no máximo digamos, como na discussão da Seção 3.5, k filhos. Portanto, se uma palavra *não* é aceita por um ABTF, então deve haver um momento a partir do qual nenhuma trajetória pode prosseguir. Isso é uma aplicação direta da idéia conhecida como *König Lemma*. Com isso, podemos provar sem dificuldade o seguinte lema que, apesar de não ser da forma de um lema de iteração, é também uma ferramenta poderosa para mostrar que certas linguagens não estão em $\mathcal{ABTF}$.

Lema 6 *Se L pertence a $\mathcal{ABTF}$, então para toda $\rho \notin L$, existe n, tal que para toda $\gamma \in \Sigma^t$, $\rho_{[1,n]} \cdot \gamma \notin L$.*

Considere o autômato da Figura 5.2, sobre o alfabeto $\Sigma = \{a\}$. Ele aceita toda palavra em que os a's estão espaçados por pelo menos uma unidade de tempo e na qual há infinitos pares de a's espaçados exatamente por uma unidade de tempo. Formalmente: $L(\mathcal{A}_4) = \{(a^\omega, \overline{\tau}) \mid \forall i [\ (\tau_{i+1} - \tau_i \geq 1) \wedge (\exists j,\ j > i,\ \tau_{j+1} - \tau_j = 1)\ ]\}$.

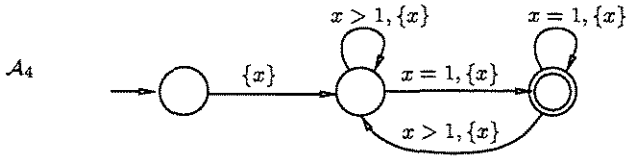

Figura 5.2: $\mathcal{ABTF}$ é subclasse própria de $\mathcal{ABT}$

Teorema 9 $L(\mathcal{A}_4) \notin \mathcal{ABTF}$.

Prova. Por contradição. Assuma que $\mathcal{B} = \langle \Sigma, Q, Q_0, X, T, F \rangle$ é um ABTF e que $L(\mathcal{B}) = L(\mathcal{A}_4)$. Considere a palavra temporizada $\rho^2 = (a^\omega, \overline{\tau^2})$, onde para qualquer i, $\tau_i^2 = 2i$. Assim, $\rho^2 \notin L(\mathcal{B})$.

Aplicando o Lema 6 a $L(\mathcal{B})$ e ρ^2, existe n tal que para todo $\gamma \in \Sigma^t$, $\rho^2_{[1,n]} \cdot \gamma \notin L(\mathcal{B})$. Mas tome qualquer $\rho \in L(\mathcal{A}_4)$. Por construção de ρ^2, nós temos $\rho^2_{[1,n]} \cdot \rho \in L(\mathcal{A}_4)$, uma contradição. □

5.1 U_{ABTF} é Π^0_1-completo

Para mostrar a pertinência de U_{ABTF} em Π^0_1 precisamos notar apenas que, pelo Lema 6, dado um ABTF A e uma palavra ρ, se para todo i existe uma trajetória de A sobre $\rho_{[1,i]}$, então tem que existir uma trajetória infinita de A sobre ρ, e que será de aceitação pois A é um ABTF.

Teorema 10 $U_{ABTF} \in \Pi^0_1$

Prova. Seja $\mathcal{A}_0, \mathcal{A}_1, \ldots$ uma indexação recursiva de todos os ABTFs. Seja d_6 uma função mapeando números naturais a palavras temporizadas finitas. Considere uma máquina de Turing M_{H_5} que, dada uma tupla $\langle i, z \rangle \in \mathbb{N}^2$, tem o seguinte comportamento:

1. Decodifica i de acordo com d_6 obtendo $\rho = (\overline{\sigma}, \overline{\tau})_k$;
2. Obtém $\mathcal{A}_z = \langle \Sigma, Q, Q_0, X, T, Q \rangle$ a partir de z;
3. Simula A_z sobre ρ. Se existe uma trajetória de A_z sobre ρ, então **aceita**; caso contrário **rejeita**.

Então $U_{\text{ABTF}} = \{z \mid \forall i \, H_5(i, z)\}$, como é fácil verificar. □

Informalmente, podemos dizer que a Π^1_1-dificuldade de U_{ABT}, U_{ABTQD} e U_{ABTMP} é devida ao fato de que estes autômatos podem reconhecer qualquer palavra temporizada que *não* representa uma computação infinita de uma máquina de Turing de 2 contadores. Já um ABTF não tem essa capacidade, pois $U_{\text{ABTF}} \in \Pi^0_1$. Entretanto, um ABTF pode reconhecer qualquer palavra que *não* representa uma computação finita de uma máquina de 2 contadores.

Teorema 11 *U_{ABTF} é Π_1^0-difícil.*

Prova. Para qualquer $C \in \Sigma_1^0$, existe uma relação recursiva H_C tal que $C = \{x \in \mathbb{N} \mid \exists n\, H_C(n,x)\}$. Dado $x \in \mathbb{N}$ nós construímos um ABTF A_x tal que A_x é universal se e somente se $x \notin C$.

Seja M_x uma máquina de Turing (não necessariamente não-determinística) que pára, sobre a fita vazia, se e somente se $x \in C$. Esta máquina precisa apenas simular M_{H_C} sobre $\langle i, x\rangle$ para valores crescentes de i, parando se M_{H_C} aceita para algum i. Aqui, M_{H_C} é a máquina subjacente à relação H_C.

A máquina M_x pode ser simulada por uma máquina de Turing de 2 contadores, como descrito na Seção 2.3.2, com a adição de um tipo de instrução: (g) pare. Uma computação finita desse tipo de máquina é uma seqüência finita de configurações relacionadas terminando numa configuração com a instrução "pare". Assuma, sem perda de generalidade, que somente a última instrução da máquina, a k-ésima instrução, é do tipo (g). Defina a linguagem temporizada L sobre o alfabeto $\{b_1, b_2, \ldots, b_k, a_1, a_2\}$ tal que $(\overline{\sigma}, \overline{\tau}) \in L$ se e somente se:

1. $\overline{\sigma} = b_{i_1} a_1^{c_1} a_2^{d_1} b_{i_2} a_1^{c_2} a_2^{d_2} \ldots b_{i_n} a_1^{c_n} a_2^{d_n} \ldots$, e $(i_1, c_1, d_1)(i_2, c_2, d_2)\ldots(i_n, c_n, d_n)$ é uma computação finita de M_x (assim, $i_n = k$);

2. para todo $j \leq n$, condições (a) a (d), como na Seção 2.3.2, valem.

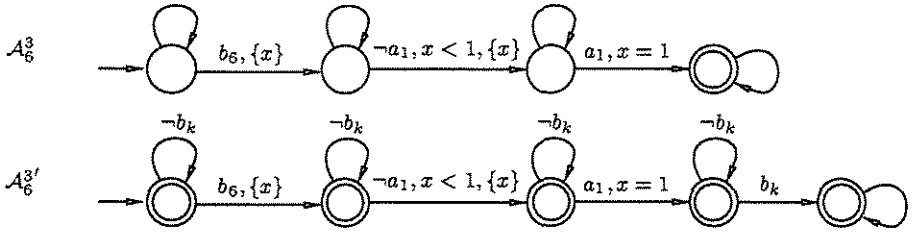

Figura 5.3: Um ABT e um ABTF que cumprem a mesma função na prova de cota inferior para universalidade

Da mesma forma que na Seção 3.3, pode ser mostrado, sem dificuldade, que a linguagem $\overline{L}$ pode ser definida como uma disjunção finita de linguagens temporizadas que podem todas ser aceitas por ABTFs. Como um exemplo, a Figura 5.3 repete o autômato $\mathcal{A}_6^3$,

da Figura 3.4, que é usado para verificar que "os a_1's não estão casados". A linguagem análoga para ABTF, aceita pelo autômato $\mathcal{A}_6^{3'}$, é $\{(\overline{\sigma},\overline{\tau}) \mid \exists i \exists j \exists k \exists \ell,\ i < j < k < \ell,\ \sigma_i = b_6, \sigma_j \neq a_1,\ \tau_j - \tau_i < 1,\ \sigma_k = a_1,\ \tau_k - \tau_j = 1,\ \sigma_\ell = b_k\}$. Note que não poderíamos simplesmente transformar todos os lugares de $\mathcal{A}_6^3$ em finais; isso tornaria o autômato universal. Todos os demais autômatos são de natureza similar e são análogos aos autômatos da Seção 3.3. □

5.2 Relacionamento entre as Classes

As classes $\mathcal{ABTQD}$, $\mathcal{ABTMP}$ e $\mathcal{ABTF}$ são todas incomparáveis, duas a duas. A única dependência é expressa pelo teorema abaixo, que mostra que não há linguagens em $(\mathcal{ABTMP} \cap \mathcal{ABTF}) \setminus \mathcal{ABTQD}$. Para qualquer outra combinação, a Seção 5.2.1 apresenta uma linguagem. O relacionamento entre as classes está ilustrado na Figura 5.4, que também indica a complexidade do problema da universalidade, sumarizando os resultados desta tese.

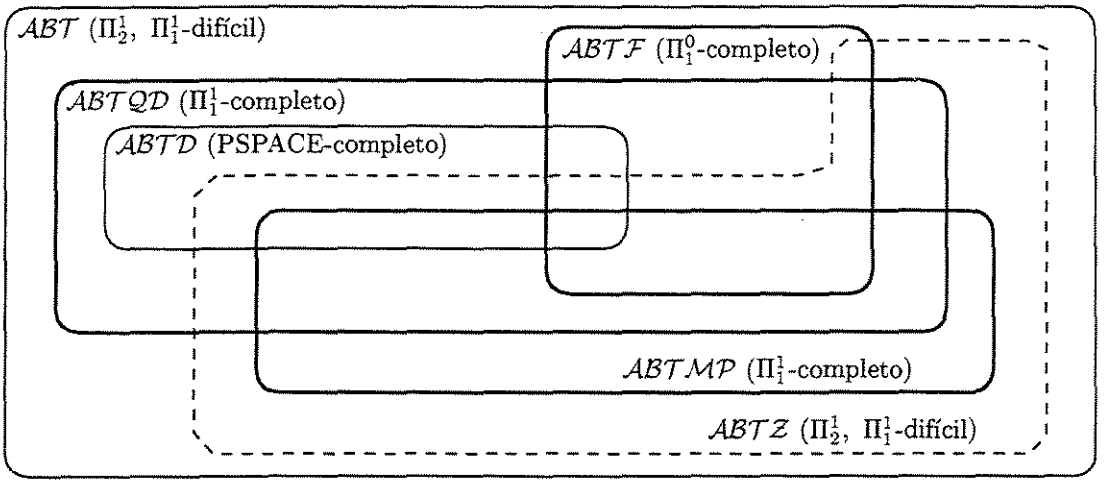

Figura 5.4: Relacionamento entre as classes de ABT

Teorema 12 *Se* $L \in (\mathcal{ABTMP} \cap \mathcal{ABTF})$, *então* $L \in \mathcal{ABTQD}$.

Prova. Seja $A_1 = \langle \Sigma, Q_1, Q_{01}, X_1, T_1, F_1 \rangle$ e $A_2 = \langle \Sigma, Q_2, Q_{02}, X_2, T_2, F_2 \rangle$ respectivamente um ABTMP e um ABTF, tal que $L(A_1) = L(A_2) = L$. Seja $G = \{q \in F_1 |$ existe

$\rho \in L(A_1)$ e existe uma trajetória $r = (\overline{q}, \overline{\nu})$ de A_1 sobre ρ, tal que $\forall i \exists j,\ j > i,\ q_j = q\}$.

Nós mostraremos que para o ABTQD A abaixo temos $L(A) = L$. O ABTQD A está ilustrado na Figura 5.5. Ele é obtido da mesma tabela temporizada de A_1, com a adição de um único estado final, como indicado. Mais rigorosamente, $A = \langle \Sigma, Q, Q_{01}, X_1, T, F \rangle$:

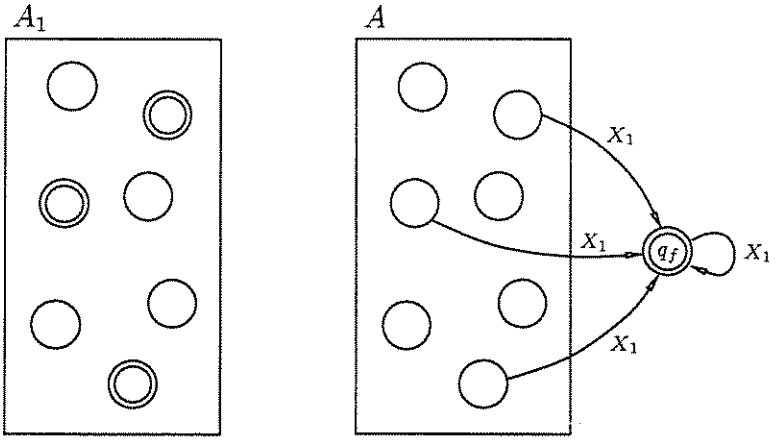

Figura 5.5: $L(A) = L(A_1)$, ABTQD A e ABTMP A_1, se $L(A_1) \in \mathcal{ABTF}$

- $Q = Q_1 \cup \{q_f\}$ (união disjunta);
- $F = \{q_f\}$;
- $T = \{\langle q, q', a, \delta, \lambda \rangle | \ [\langle q, q', a, \delta, \lambda \rangle \in T_1] \vee [q' = q_f \wedge \lambda = X_1 \wedge \delta = \texttt{true} \wedge q \in G \cup \{q_f\}]\}$.

Por construção de A, podemos verificar sem dificuldade que, se $\rho \in L(A_1)$, então $\rho \in L(A)$. Veremos agora a outra direção.

Note que qualquer trajetória de aceitação de A tem que passar ao menos uma vez por um lugar em G. Assim, se $\rho \in L(A)$, então podemos encontrar uma trajetória $r = (\overline{q}, \overline{\nu})$ de A sobre ρ tal que existe um m para o qual $q_m \in G$. Mas note que $r_{[0,m]}$ é, então, uma trajetória finita de A_1 sobre $\rho_{[1,m]}$.

Como $q_m \in G$, por definição existe $\rho' \in L(A_1)$ e existe uma trajetória $r' = (\overline{q'}, \overline{\nu'})$ de A_1 sobre ρ', tal que $\forall i \exists j,\ j > i, q'_j = q_m$. Seja m' o menor natural tal que $q'_{m'} = q_m$. Veja a ilustração na Figura 5.6. Agora seja $\rho'' = \rho_{[1,m]} \cdot \rho'_{[m')}$. Como A_1 tem a trajetória $r'' = r_{[0,m]} \cdot r'_{[m')}$ sobre ρ'', temos $\rho'' \in L(A_1)$.

Agora nós argumentamos que $\rho \in L(A_1)$. Suponha, para efeito de contradição, que $\rho \notin L(A_1)$. Como $L(A_1) = L(A_2)$, e A_2 é um ABTF, pelo Lema 6, existe n, tal que

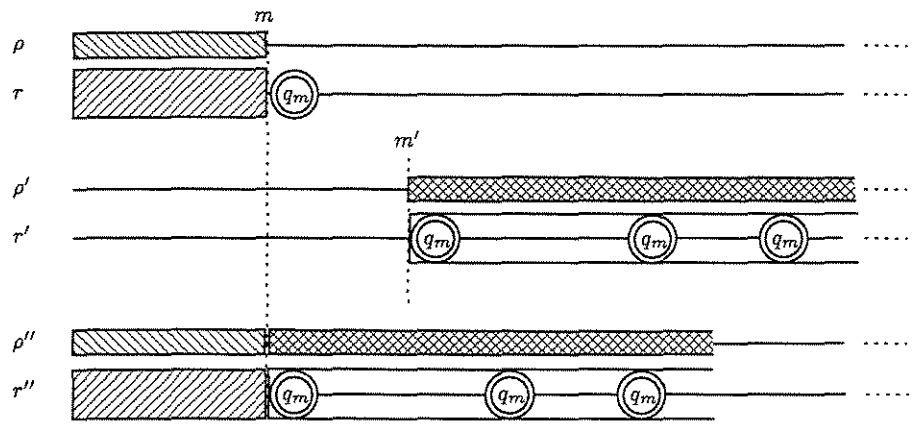

Figura 5.6: Demonstração de que $(\mathcal{ABTMP} \cap \mathcal{ABTF}) \subset \mathcal{ABTQD}$

para todo $\gamma \in \Sigma^+$, $\rho_{[1,n]} \cdot \gamma \notin L(A_1)$. Se $n \leq m$, então nós já temos uma contradição, pois $\rho'' = \rho_{[1,n]} \cdot \rho_{[n+1,m]} \cdot \rho'_{[m')}$. Se $n > m$, então, como $\rho_{[1,m]} = \rho''_{[1,m]}$, pelo Lema 3, existe $\ell > n$, tal que $\rho''' = \rho_{[1,\ell]} \cdot \rho''_{[\ell+1)} \in L(A_1)$. Novamente uma contradição, pois $\rho''' = \rho_{[1,n]} \cdot \rho_{[n+1,\ell]} \cdot \rho''_{[\ell+1)}$. □

	L_0	L_1	L_2	L_3	L_4	L_5	L_6	L_7	L_8	L_9	L_{10}
$\mathcal{ABTF}$	√			√					√	√	√
$\mathcal{ABTMP}$	√	√	√					√		√	
$\mathcal{ABTD}$	√				√			√	√		
$\mathcal{ABTQD}$	√	√			√	√		√	√	√	√

Tabela 5.1: O catálogo de linguagens temporizadas

5.2.1 Catálogo de Linguagens

A lista seguinte apresenta uma linguagem temporizada para cada interseção não-vazia de classes na Figura 5.4. A Tabela 5.1 indica com uma marca "√" exatamente as classes nas quais cada linguagem está contida. Para cada linguagem, uma prova de que ela *não* está contida numa determinada classe, quando indicado na tabela, pode ser obtida, sem dificuldade, com uma combinação das idéias apresentadas nas provas dos Teoremas 2, 7 e 9. Por serem repetitivos, detalhes desses argumentos são omitidos. Para cada linguagem, é

possível encontrar um ABT para cada classe indicada na tabela, que a aceita. A figura 5.7 dá cinco exemplos, onde $L(A_i) = L_i$, L_i como indicado na lista. Os outros autômatos são similares.

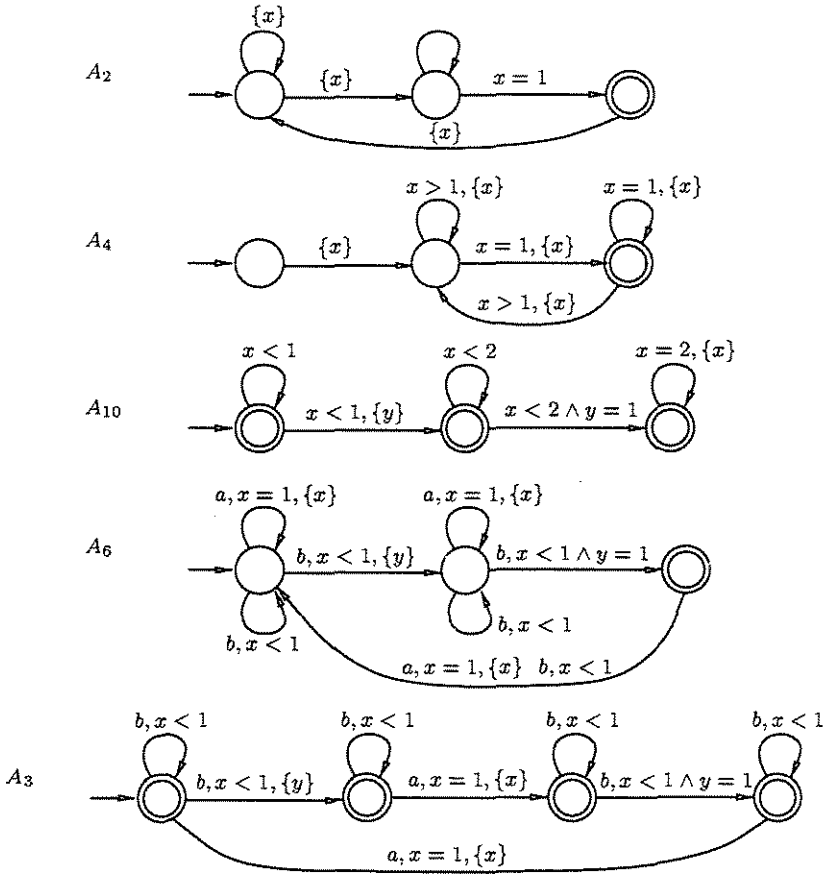

Figura 5.7: Alguns exemplos de ABT

- $L_0 = \Sigma^t$ (linguagem universal);
- $L_1 = \{(a^\omega, \overline{\tau}) | \exists i \exists j,\ j > i,\ \tau_j = \tau_i + 1\}$;
- $L_2 = \{(a^\omega, \overline{\tau}) | \forall k \exists i \exists j,\ j > i > k,\ \tau_j = \tau_i + 1\}$;
- $L_3 = \{(\overline{\sigma} \in (b^+a)^\omega, \overline{\tau}) | \forall i\ [\ (\tau_i \in \mathbb{N} \Leftrightarrow \sigma_i = a) \wedge (\ \exists j,\ \tau_j = i) \wedge (\exists j \exists k,\ 2i < \tau_j < 2i + 1,\ \tau_k = \tau_j + 1)\ ]\}$;
- $L_4 = \{(a^\omega, \overline{\tau}) | \forall i\ [\ \tau_{i+1} - \tau_i \geq 1 \wedge \exists j,\ j > i,\ \tau_{j+1} - \tau_j = 1\ ]\}$;

- $L_5 = \{(\overline{\sigma} \in (a+b)^\omega, \overline{\tau}) | \exists i \forall j,\ j > i \Rightarrow \sigma_j = b\}$;
- $L_6 = \{(\overline{\sigma} \in (b^+a)^\omega, \overline{\tau}) | \forall i\ [\ (\tau_i \in \mathbb{N} \Leftrightarrow \sigma_i = a) \wedge (\exists j,\ \tau_j = i) \wedge (\exists j \exists k,\ k > j > i,\ \tau_k = \tau_j + 1,\ \sigma_j = \sigma_k = b)\ ]\}$;
- $L_7 = \{(\overline{\sigma} \in (b^+a)^\omega, \overline{\tau}) | \forall i \exists j,\ j > i,\ \sigma_j = a\}$;
- $L_8 = \{(a^\omega, \overline{\tau}) | \forall i,\ \tau_i = i\}$;
- $L_9 = \{(a^\omega, \overline{\tau}) | \exists i \exists j,\ \tau_j = \tau_i + 1,\ \tau_j < 2\}$;
- $L_{10} = \{(a^\omega, \overline{\tau}) | [\exists i \exists j,\ \tau_j = \tau_i + 1,\ \tau_j < 2] \wedge [\exists i((\tau_i = 2) \wedge (\forall j,\ j \geq i \Rightarrow \tau_{j+1} = \tau_j + 2))]\}$.

Capítulo 6

Conclusões

O problema da Universalidade para Autômatos Büchi Temporizados Não-determinísticos é indecidível mas resiste a uma caracterização simples do seu grau de indecidibilidade. Por um lado, essa classe de autômatos parece não ter, como foi discutido na Introdução e nos capítulos anteriores, expressividade suficiente para que seu problema da universalidade seja Π_2^1-completo. Por outro lado, como resultado desta tese, tem *mais* expressividade do que algumas subclasses que são Π_1^1-completas. Essa idéia de relacionar a complexidade do problema da universalidade com a expressividade dos autômatos está ilustrada na Figura 6.1, para o caso de $\mathcal{ABTQD}$. A situação é interessante pois intuitivamente se espera que o problema seja completo para a classe Π_2^1 ou para a classe Π_1^1, ao mesmo tempo que se espera que a curva na figura seja estritamente crescente.

Não há, como já dissemos, uma métrica para a expressividade, nem uma proposta de tornar essa argumentação mais rigorosa. O que podemos afirmar é que a curva não pode ser decrescente[1]. Mas não há, em princípio, impedimento para que o grau seja o mesmo para duas classes de expressividade distintas e comparáveis. É possível produzir, por exemplo, uma superclasse própria da classe $\mathcal{ABTQD}$, com universalidade Π_1^1-completo. Poderíamos tomar uma linguagem L qualquer fora de $\mathcal{ABTQD}$ (portanto L não poderia ser a linguagem universal), encontrar um autômato A_L que a aceite e definir uma nova classe de autômato: $ABTQD+L$, composto de todos os ABTQDs mais A_L. Essa superclasse trivial teria universalidade Π_1^1-completo. Contra essa abordagem de produzir

[1] É preciso, também, assumir que a transformação entre os formalismos é efetiva, ou seja, que o problema da universalidade para o menos expressivo pode ser efetivamente reduzido ao problema da universalidade para o mais expressivo.

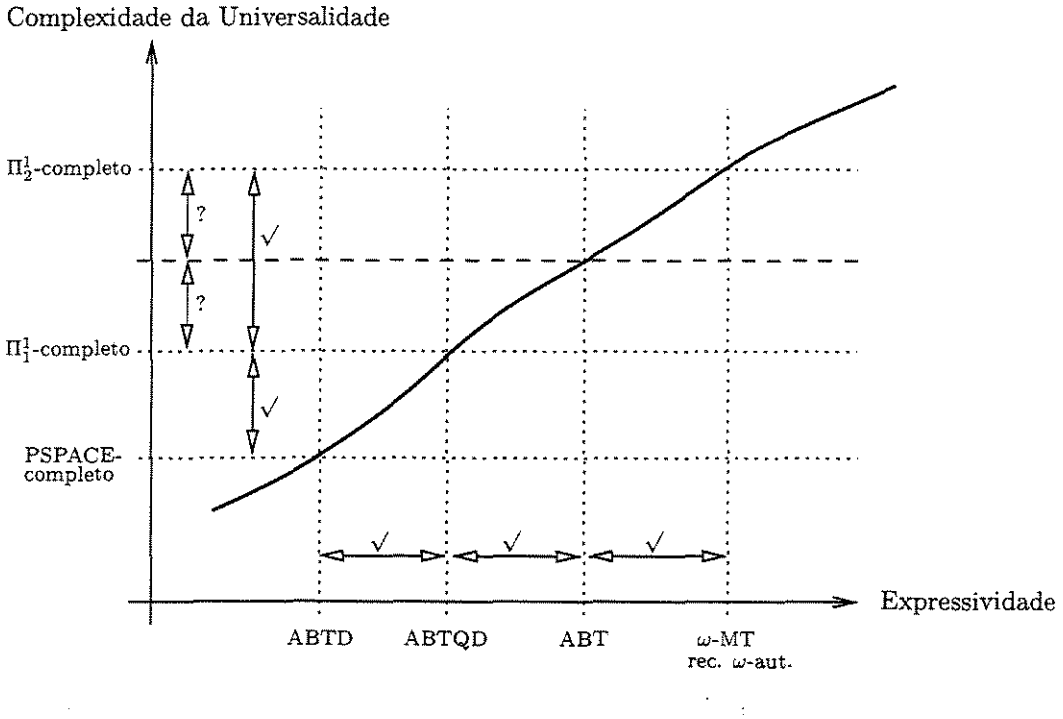

Figura 6.1: Expressividade versus Complexidade da Universalidade

classes triviais estariam: a possibilidade de que não fossem robustas como as definidas nesta tese, no sentido de não apresentarem as mesmas propriedades de fechamento de $\mathcal{ABT}$; e a dificuldade de conciliar uma tentativa de extensão da classe, adicionando talvez uma família infinita de autômatos para buscar a robustez, com a necessidade de manter a capacidade de indexação recursiva e a Π_1^1-completude da universalidade. Outra alternativa seria considerar, por exemplo, a união $\mathcal{ABTQD} \cup \mathcal{ABTMP}$ como sendo uma nova classe. Certamente essa união é superclasse própria tanto de $\mathcal{ABTQD}$ quanto de $\mathcal{ABTMP}$. Porém, nenhuma nova linguagem seria acrescentada e restaria o exercício de demonstrar a robustez da classe união.

A nossa investida mais promissora de obter uma superclasse própria mais geral e razoável, ainda Π_1^1-completa, para alguma das classes apresentadas, foi a classe $\mathcal{ABTZ}$. Como vimos, porém, esbarramos na dificuldade de obter um algoritmo Π_1^1. A mesmas dificuldades encontradas para $\mathcal{ABT}$ já se apresentam para $\mathcal{ABTZ}$, uma subclasse própria. De qualquer forma, qualquer saída para o problema em aberto resultará numa figura, similar à Figura 5.4, bastante interessante.

Temporização faz realmente a diferença? Na literatura, pelo menos quando a motivação é alguma aplicação prática, há discussões sobre a natureza e necessidade da temporização explícita. Boas referências para essa discussão são [6, 7]. Quer dizer, a pergunta é se o tempo não é apenas mais uma variável do sistema que, na verdade, não adiciona uma dimensão de natureza diferente. No caso do nosso problema da universalidade vale a pena notar o seguinte. Nenhuma das três restrições sintáticas que deram origem às classes $\mathcal{ABTQD}$, $\mathcal{ABTMP}$ e $\mathcal{ABTF}$ é de natureza, ou de motivação, temporal. Elas se aplicam a qualquer tipo de ω autômato. Só que, para ABT em particular, e combinando com a parte temporizada dos autômatos, provocaram uma simplificação na estrutura das trajetórias dos autômatos sobre as palavras, que permite a obtenção de algoritmos para as hierarquias. Poderia ser levantado o argumento de que talvez seja exatamente isso que falta às tentativas de resolver a questão em aberto. Talvez a consideração de algum aspecto particular da estrutura temporizada dos autômatos é que seria a chave para solucionar a questão. Nossa experiência, porém, pelo menos intuitivamente, indica que a essência do problema não está na parte temporizada dos autômatos. Ou melhor, a temporização certamente é necessária para a complexidade da estrutura das trajetórias dos autômatos, mas a todo instante a questão parece se apresentar mais como sendo um problema geral de Sistemas de Transição Infinitos e não algo específico. Colocando de outra forma, uma solução para a questão em aberto, na nossa opinião, estaria baseada em propriedades gerais de sistemas não-determinísticos mais do que em questões sobre tempo-real.

Histórico e Referências Adicionais. A idéia inicial do projeto desta tese era trabalhar com verificação de Sistemas Híbridos Probabilísticos. No início, estudamos boa parte da literatura sobre sistemas híbridos e de tempo-real, sobre verificação probabilística e também alguns algoritmos e ferramentas práticas disponíveis. No decorrer desse estudo é que surgiu e veio para o primeiro plano a questão que foi deixada em aberto no artigo de Alur e Dill [4], que acabou se tornando o centro da tese. O interesse por essa questão aumentou com o tempo; e começamos a perceber e explorar subclasses interessantes de $\mathcal{ABT}$, com o propósito de atacar a questão.

Nesse ínterim, a componente probabilística do projeto, de motivação prática, foi deixada em segundo plano. O que concluímos com o estudo dessa componente é que a adição

de probabilidades a sistemas híbridos, e mesmo a sistemas de tempo-real, aumenta em demasia a complexidade computacional da verificação. Além disso, a comunidade de verificação parece ainda estar concentrada em resolver os problemas que surgem na prática para sistemas não-probabilísticos.

Entretanto, antes de desviar a atenção da parte probabilística, obtivemos alguns resultados, também de relevância primordialmente teórica, sobre a verificação de Sistemas de Tempo-Real Probabilísticos contra propriedades modeladas com ABTs. O apêndice A traz a versão Relatório Técnico de um artigo que foi publicado em [43]. A ligação deste trabalho com a tese apresentada é o não-determinismo. Neste artigo a essência também é contornar o não-determinismo para decidir se o sistema probabilístico exibe ou não um certo comportamento. A idéia central do algoritmo, *relógios genéricos*, pode ser estendida para a verificação não-probabilística, que recai, como foi comentado na Introdução, no problema da Inclusão de Linguagens. O apêndice B traz um relatório técnico com detalhes dessa extensão.

Apêndice A

Não-determinismo e Verificação Probabilística

On the Verification of Nondeterministic Automata Specifications of Probabilistic Real-Time Systems

Arnaldo V. Moura and Guilherme A. Pinto
{arnaldo,guialbu}@dcc.unicamp.br

Abstract

In [2], Alur et al. presented an algorithm for the problem of verifying deterministic timed automata specifications of probabilistic real-time systems given as generalized semi-Markov processes; and posed the question of the verification of nondeterministic specifications. We give a partial answer to the question, extending their method so that processes, with a fairly acceptable restriction, can be tested against any timed automaton. These include, for instance, real-time models of digital circuits where the key property is that the delay distributions have non-zero lower bounds.

A.1 Introduction

Techniques for automatic verification of timing properties of non-probabilistic real-time systems have already been widely studied [7, 4, 28] and applied to practical problems. In many cases, however, one may want to consider the probabilistic behavior of a physical system—in these cases, the best a verification method can do is to say that the system satisfies the property with probability one. Typically, system models based on state-transition graphs can account for probabilities either only in the discrete transitions, or also in the delays of the events triggering the discrete transitions. In the latter case, when there are continuous probability distributions in the delays, there are, at least, three approaches.

In [1], Alur et al. presented a model-checking algorithm for timed computation tree logic (TCTL) formulas, where the existential and universal path quantifiers are interpreted as "with positive probability" and "with probability one" respectively, so that the verification is qualitative (in what concerns probability bounds). The system model defines a generalized semi-Markov process.

Recently, Kwiatkowska et al. [33] proposed a procedure to check systems, given by a similar model, against formulas of the probabilistic timed computation tree logic (PTCTL), which is a version of TCTL where the path quantifiers are amended with quantitative probabilistic bounds, so that the verification is quantitative, although (unavoidably) approximative.

These logics, however, are purely propositional, and do not have the counting ability, for example, of the popular timed automata (TA) formalism [28]. There are certain interesting properties that can be specified by TA, and such that TCTL or PTCTL cannot describe. An example is the *convergent bounded response* property: "two events a and b alternate and eventually always the time difference between a and the next b is less than or equal to 2 seconds" [4, 2]. Thus, in [2], Alur et al. extended their previous method to the verification of deterministic TA specifications, which can express the convergent bounded response property, for instance; and posed the question of the verification of nondeterministic TA specifications. Besides the fact that nondeterminism facilitates the specification of properties and gives rise to, potentially, smaller models, the interest in this question also stems from the fact that nondeterministic TA are strictly more expressive than deterministic TA [4]. Unfortunately, the probabilistic verification problem, which is

commonly solved by complementing the specification, resembles the language inclusion problem, which is undecidable for nondeterministic TA [4]. Nondeterministic TA are not closed under complementation [4], so that one must resort to some other technique to cope with the nondeterminism in the specification. For instance, in the discrete time domain, where the system is a Markov chain and the specification an ω-automaton, Courcoubetis and Yannakakis [16] solved a more general problem with the usual subset construction. But, for TA, it was not clear in [2] how one could apply such a construction, and also carry through the probabilistic analysis.

In this paper, we show that a subset construction on the states (location plus valuation for the clocks) of the automaton admit the definition of the integral parts/order of fractional parts equivalence relation over the set of states, so that the method in [2] can be extended in a uniform way until the point where only probabilistic information are left. At this point, the method constructs a graph, over which the probabilistic analysis is done. When we apply the subset construction, for some instances, this graph is infinite, and the method cannot be applied—the process and the automaton may synchronize is such a way that an unbounded amount of information is needed for the method to succeed. Nevertheless, we show that for fairly acceptable restrictions on the process model, the graph is guaranteed to be finite, in such a way that the powerful nondeterministic timed automata formalism can indeed be used to specify properties. For example, if there is a bound on the number of events generated by the process, in a time interval of unit length, then our method guarantees that it can be tested against any TA, be it deterministic or nondeterministic.

The paper is organized as follows. Section A.2 explains the system model of probabilistic real-time processes. In Section A.3 we review the specification formalism of timed automata and define the probabilistic verification problem. Section A.4 presents the construction, gives an example for which the method cannot be used, and shows that most practical instances *are* solvable. Section A.5 concludes with some final remarks.

A.2 The System Model

Our system model is inspired on the one defined in [2], and we use the same notation. Informally, the system moves probabilistically through a finite set of states S according

to a sequence of discrete events. The sojourn time in a state and the event triggering the probabilistic transition are chosen in such a way that it defines a *generalized semi-Markov process* [49] over the set S. We first give the formal definition and then discuss the operation of the process.

Definition 1 We say that a probability distribution is *bounded* if it has support on a bounded interval $[t_1, t_2]$, t_1 and t_2 in $\mathbb{Q}$ (the set of non-negative rational numbers), and probability density function f, such that, if $t_1 = t_2$, then $f(t_1) = 1$ (discrete probability distribution), and if $t_1 < t_2$, then $\int_{t_1}^{t_2} f(t)dt = 1$ (continuous probability distribution).

We denote by D the set of all bounded and exponential distributions. The algorithm needs only trivial modifications to treat bounded distributions with support of k bounded intervals, for any k in $\mathbb{N}$ (the set of natural numbers).

Definition 2 A *real-time probabilistic process* $\mathcal{M}$ is a tuple $\langle S, E, act, f_0, next, dist \rangle$, where

- S is a finite set of states;
- E is a finite set of basic events. The set of all nonempty subsets of E is denoted by Δ. An element $a \in \Delta$ is called an event of $\mathcal{M}$;
- $act : S \to \Delta$ is a function associating a set of basic events to each state, the *active* basic events of a state;
- $f_0 : S \to [0, 1]$ is a initial distribution function such that $\sum_{s \in S} f_0(s) = 1$;
- $next : S \times \Delta \times S \to [0, 1]$ is a probabilistic transition function. For every state s and every $a \in \Delta$, $\sum_{s' \in S} next(s, a, s') = 1$;
- $dist : E \to D$ is a function associating each basic event $x \in E$ to a probability distribution in D. Given *dist*, we denote by E_e and E_b, respectively, the set of basic events with exponential distribution, and the set of basic events with bounded distributions; and distinguish two functions, $l : E_b \to \mathbb{Q}$ and $u : E_b \to \mathbb{Q}$, giving the lower and upper bounds of the distributions, respectively. We call $x \in E_b$ a *fixed-delay* basic event if $l_x = u_x$, and a *variable-delay* basic event if $l_x < u_x$. Let

$act_e(s) = act(s) \cap E_e$ and $act_b(s) = act(s) \cap E_b$. Also, we write f_x for the probability density function associated to an exponential or variable delay basic event x.

The process works as follows. Each basic event $x \in E$ has an associated clock, also referred to as x. The process starts in some state s such that $f_0(s) > 0$. For each $x \in act(s)$, a value d_x is randomly and independently chosen from the distribution $dist(x)$ and the clock x is set to $-d_x$. The values of all the clocks increase with the real time until some clock reaches 0, when, then, a state transition occurs. Let $a \in \Delta$ be the set of basic events reaching value 0 in s. The process moves to some state s' such that $next(s, a, s') > 0$. The reading of the clocks in $act(s')$ are obtained as follows:

- Let $old(s, a, s') = act(s') \cap (act(s) \setminus a)$. The basic events in $old(s, a, s')$ are active in s and s'—the value of their clocks is not modified;
- Let $new(s, a, s') = act(s') \setminus old(s, a, s')$. Each clock x in $new(s, a, s')$ is set to $-d_x$, where d_x is, again, randomly and independently chosen from $dist(x)$.

Example 1 Consider the process $\mathcal{M}_s$ in Fig. A.1, where $act(s_1) = \{a, c\}$, $act(s_2) = \{b\}$ and $act(s_3) = \{d\}$. It models two concurrent basic events a and c, with uniform distribution between 0 and 4, and between 1 and 3, respectively. The process starts in state s_1 and, if a happens before c, then the process goes to s_2 with probability 1; c is no longer active and a response basic event b is scheduled with uniform distribution between 1 and 2. If c happens before a in s_1, then, with probability 0.2, c is simply rescheduled; and with probability 0.8, the process goes to s_3, and a response basic event d is scheduled with uniform distribution between 0 and 1. Also, when b happens in s_2, with a small probability, the process goes to s_3. Note that, the probability that a and c happen simultaneously is zero. Two basic events can happen at the same time only if they are fixed-delay. □

The process is not Markov because non-exponential distributions can be associated to the delays, and is not semi-Markov because not even at the transition times the Markov property holds, since not all events in $act(s)$ are rescheduled when the process enters s. We see that $\mathcal{M}$ indeed defines a generalized semi-Markov process over S because we can retrieve the Markov property by allowing a generalized notion of state. A *generalized state* of $\mathcal{M}$ has the form $\langle s, v \rangle$, where $s \in S$ and v is a mapping from $act(s)$ to the

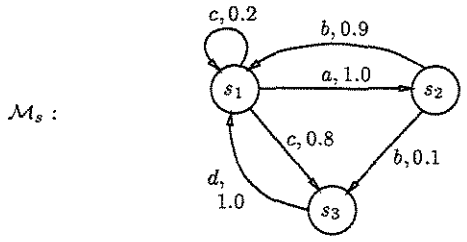

Figura A.1: A real-time probabilistic process $\mathcal{M}_s$

non-positive reals, that is, a particular reading for each active clock, which we call a *clock interpretation for* E. If we now let the state space be the space of all generalized states, then we have a Markov process Y. For $t \in \mathbb{R}$ (the set of non-negative real numbers), we write $v + t$ for the clock interpretation which maps every clock x to $v(x) + t$.

A.3 Timed Automata and the Verification Problem

Timed automata were proposed in [4] as a formalism for the verification of real-time systems. Informally, a timed automaton is an ω-automaton together with a finite set of clock variables whose values increase with the real time. Every transition of the automaton has a constraint on the values of the clocks and can be taken only if the clocks satisfy the constraint. In addition, a transition may reset some of the clocks. As we will see, timed automata accept timed words instead of ω-words.

Definition 3 A *timed word* ρ over a finite alphabet Σ is a pair (σ, τ) where

- $\sigma = \sigma_1\sigma_2\cdots$ is a sequence of symbols $\sigma_i \in \Sigma$ (an ω-word over Σ);
- $\tau = \tau_1\tau_2\cdots$ is an strictly increasing sequence of time values $\tau_i \in \mathbb{R}$, $\tau_i > 0$, satisfying the *progress* property: for every $t \in \mathbb{R}$, there is some $i \geq 1$ such that $\tau_i > t$. Given $c \in \mathbb{R}$, we write $c \cdot \tau$ for the sequence obtained from τ by multiplying every τ_i by c.

In a timed word (σ, τ), the time value τ_i is interpreted as the occurrence time of the event σ_i. For example, the following is the initial prefix of a timed word ρ_s over $\{a, b\}$: $(a, 3.1) \to (b, 6) \to (a, 6.2) \to (b, 7.5) \to \cdots$.

Definition 4 Given a finite set X of clock variables, a *clock constraint* δ over X is defined inductively by $\delta := x \leq c \,|\, c \leq x \,|\, \neg\delta \,|\, \delta_1 \wedge \delta_2$, where $x \in X$ and $c \in \mathbb{Q}$. The set of all clock constraints over X is denoted by $\Phi(X)$.

Definition 5 A *timed (Büchi) automaton* $\mathcal{A}$ is a tuple $\langle \Sigma, Q, Q_0, X, T, F \rangle$, where

- Σ is a finite alphabet;
- Q is a finite set of locations;
- $Q_0 \subseteq Q$ is a set of start locations;
- X is a finite set of clocks;
- $T \subseteq Q \times Q \times \Sigma \times 2^X \times \Phi(X)$ is a set of transitions. For a transition $\langle q, q', a, \lambda, \delta \rangle$ from location q to location q', on symbol a, δ gives the constraint to be satisfied and λ the set of clocks to be reset;
- $F \subseteq Q$ is a set of accepting locations.

The operation of the automaton $\mathcal{A}$ is obtained by defining runs of $\mathcal{A}$ over timed words. For this, let a *clock interpretation for* X be a mapping from X to $\mathbb{R}$, that is, a particular reading of the clocks in X. A *generalized location* of $\mathcal{A}$ has the form $\langle q, \nu \rangle$, where $q \in Q$ and ν is a clock interpretation for X. For $t \in \mathbb{R}$, we write $\nu + t$ and $t \cdot \nu$, respectively, for the clock interpretation which maps every clock x to $\nu(x) + t$ and to $t \cdot \nu(x)$. A clock interpretation ν for X satisfies a clock constraint δ over X iff δ evaluates to true when each clock x is replaced by $\nu(x)$. Given a transition $\langle q, q', a, \lambda, \delta \rangle$ and a generalized location $\langle q, \nu \rangle$, the transition is said to be *enabled* if ν satisfies δ.

Definition 6 A *run* $r = (\overline{q}, \overline{\nu})$, of a timed automaton $\mathcal{A}$ over a timed word $\rho = (\sigma, \tau)$ is an infinite sequence of generalized locations of the form

$$r : \langle q_0, \nu_0 \rangle \xrightarrow[\tau_1]{\sigma_1} \langle q_1, \nu_1 \rangle \xrightarrow[\tau_2]{\sigma_2} \langle q_2, \nu_2 \rangle \xrightarrow[\tau_3]{\sigma_3} \cdots,$$

satisfying:

- *Initiation*: $q_0 \in Q_0$, and $\nu_0(x) = 0$ for all $x \in X$;

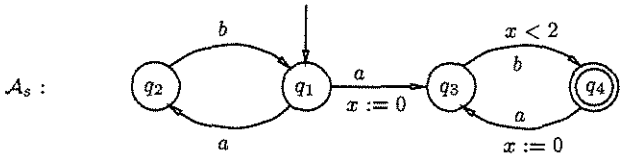

Figura A.2: A nondeterministic timed automaton $\mathcal{A}_s$

- *Consecution*: for all $i \geq 1$, there exists $\langle q_{i-1}, q_i, \sigma_i, \lambda_i, \delta_i \rangle \in T$ such that $(\nu_{i-1} + \tau_i - \tau_{i-1})$ satisfies δ_i, and $\nu_i(x) = 0$ if $x \in \lambda_i$ and $\nu_i(x) = \nu_{i-1} + \tau_i - \tau_{i-1}$ otherwise ($\tau_0 = 0$, by definition).

Given a run $r = (\overline{q}, \overline{\nu})$ over a timed word $\rho = (\sigma, \tau)$, let $inf(r)$ be the set of locations such that $q \in inf(r)$ iff $q = q_i$ for infinitely many $i \geq 1$ in r. The run r over ρ is called an *accepting run* iff $inf(r) \cap F \neq \emptyset$. Finally, the language accepted by $\mathcal{A}$ is $L(\mathcal{A}) = \{(\sigma, \tau) \mid \mathcal{A} \text{ has an accepting run over } (\sigma, \tau)\}$.

Example 2 The automaton $\mathcal{A}_s$ in Fig. A.2 expresses the convergent bounded response property mentioned in the introduction. The single clock x is reset only in the transitions from q_1 to q_3 and from q_4 to q_3. The transition from q_3 to q_4 is the only one with a clock constraint different than *true*. Note that the automaton is nondeterministic, since the transitions from q_1 to q_3 and from q_1 to q_2, which are on the same symbol, can be simultaneously enabled.

The language accepted by $\mathcal{A}_s$ is $L(\mathcal{A}_s) = \{((ab)^\omega, \tau) \mid \text{ there is } i \text{ such that for all } j \geq i,\ (\tau_{2j} < \tau_{2j-1} + 2)\}$. The following are the two possible initial prefixes of runs of $\mathcal{A}_s$ over ρ_s ($x(d)$ means $x = d$):

$$\langle q_1, x(0)\rangle \xrightarrow[3.1]{a} \langle q_2, x(3.1)\rangle \xrightarrow[6]{b} \langle q_1, x(6)\rangle \xrightarrow[6.2]{a} \langle q_2, x(6.2)\rangle \xrightarrow[7.5]{b} \langle q_1, x(7.5)\rangle \cdots,$$

$$\langle q_1, x(0)\rangle \xrightarrow[3.1]{a} \langle q_2, x(3.1)\rangle \xrightarrow[6]{b} \langle q_1, x(6)\rangle \xrightarrow[6.2]{a} \langle q_3, x(0)\rangle \xrightarrow[7.5]{b} \langle q_4, x(1.3)\rangle \cdots.$$

□

We note that timed Büchi automata have the same expressiveness as timed Muller automata [4]. In [2], Muller acceptance condition was used instead, since deterministic timed Muller automata are strictly more expressive than deterministic timed Büchi automata.

A.3.1 The Verification Problem

An instance of the verification problem is composed by a process $\mathcal{M}$ describing the probabilistic system, and a timed automaton $\mathcal{A}$ giving the specification. A particular behavior of the process $\mathcal{M}$ is described as a timed word over Δ, which gives the sequence of events and their occurrence times. From now on, we assume that the set of events Δ of $\mathcal{M}$ equals the alphabet Σ of the timed automaton $\mathcal{A}$. Let Σ^t denote the set of all timed words over Σ. The process $\mathcal{M}$ induces a probability measure[1] over Σ^t. We say that $\mathcal{M}$ satisfies $\mathcal{A}$ iff $L(\mathcal{A})$ has measure 1 in the probability measure induced by $\mathcal{M}$. In other words, the verification problem is to decide, whether or not, the probability that $\mathcal{M}$ exhibits a behavior accepted by $\mathcal{A}$ equals 1.

As an example, if we add dummy transitions for the basic events c and d (of the process $\mathcal{M}_s$) in the automaton $\mathcal{A}_s$ ($\langle q, q, \ell, \emptyset, true\rangle$, for all $q \in Q$, $\ell \in \{c, d\}$), the process $\mathcal{M}_s$ satisfies $\mathcal{A}_s$.

A.3.2 Restriction to Integer Constants

Let c be a positive real constant. Given a timed automaton $\mathcal{A}$, let $\mathcal{A}^c$ denote the automaton obtained by multiplying all the constants appearing in all the clock constraints of $\mathcal{A}$ by c. There exists a one-to-one correspondence between the runs of $\mathcal{A}$ and the runs of $\mathcal{A}^c$: given a timed word $\rho = (\sigma, \tau)$, $(\overline{q}, \overline{\nu})$ is a run of $\mathcal{A}$ over ρ iff $(\overline{q}, c \cdot \overline{\nu})$ is a run of $\mathcal{A}^c$ over $(\sigma, c \cdot \tau)$ [4].

Given a process $\mathcal{M}$, let $\mathcal{M}^c$ denote the process obtained by replacing the function *dist* by $dist^c$, such that $dist^c$ replaces the functions l by l^c, u by u^c, and, for every variable delay and exponential basic event x, the function f_x by f_x^c; where l^c, u^c and f_x^c are defined, respectively, as: $l_x^c = c \cdot l_x$, $u_x^c = c \cdot u_x$ and $f_x^c(t) = f_x(t/c)/c$. These definitions guarantee that, given any interval $[t_1, t_2]$, $\int_{t_1}^{t_2} f_x(t)\,dt = \int_{c \cdot t_1}^{c \cdot t_2} f_x^c(t)\,dt$. In particular, $\int_{l_x}^{u_x} f_x(t)\,dt = \int_{l_x^c}^{u_x^c} f_x^c(t)\,dt = 1$. Thus, we can obtain the probability measure induced by $\mathcal{M}^c$ on Σ^t from the one induced by $\mathcal{M}$ simply mapping each timed word (σ, τ) to $(\sigma, c \cdot \tau)$. The following lemma holds.

Lemma 1 *$\mathcal{M}$ satisfies $\mathcal{A}$ iff $\mathcal{M}^c$ satisfies $\mathcal{A}^c$.* □

[1] A formal definition of a similar probability measure can be found in [33]

The invariance under multiplication by a constant shows that we can restrict our attention to instances with integer constants. Given a process $\mathcal{M}$ and an automaton $\mathcal{A}$, we can choose c to be the least common multiple of the denominators of all constants appearing in the clock constraints of $\mathcal{A}$ and all constants in the ranges of the functions l and u of $\mathcal{M}$, and then, use $\mathcal{M}^c$ and $\mathcal{A}^c$ instead, which have only integer constants.

A.4 The Algorithm

Let $\mathcal{M}$ be a real-time probabilistic process, and let $\mathcal{A}$ be a timed automaton, both with integer constants. We assume that from every location $q \in Q$ of $\mathcal{A}$ and every clock interpretation ν, there is an edge $\langle q, q', a, \lambda, \delta \rangle$, for every $a \in \Sigma$, for some q', λ and δ, such that ν satisfies δ. This can be achieved by a simple transformation: add a dummy location q_d to Q and, then, transitions $\langle q, q_d, a, X, true \rangle$, for all $a \in \Sigma$, and all $q \in Q$. The set F remains unchanged. Clearly, $L(\mathcal{A})$ is not altered.

In order to cope with the nondeterminism in $\mathcal{A}$, we use the standard idea of a subset construction, applied on the generalized locations of $\mathcal{A}$. As in [2], given Y, we define an extended Markov process Y^* that simulates the runs of $\mathcal{A}$ over the behavior of $\mathcal{M}$. The process Y^* records in its states, in addition to the state of Y, a finite set of generalized locations of $\mathcal{A}$. Let A be the set of all generalized locations of $\mathcal{A}$ and let 2^A_{fin} denote the set of all finite subsets of A. Then, a state of Y^* has the form $\langle s, \upsilon, p \rangle$, where $\langle s, \upsilon \rangle$ is a state of Y and $p \in 2^A_{fin}$. The states of Y^* are updated as follows:

- *Initial states*: all states of the form $\langle s, \upsilon, p \rangle$, where $f_0(s) > 0$ and for all $x \in act(s)$, $\upsilon(x)$ is according to the probability distribution $dist(x)$ and $p = \{\langle q, \nu \rangle \mid q \in Q_0 \text{ and } \nu(x) = 0 \text{ for all } x \in X\}$;

- *Time-passage states*: if $Y^*_t = \langle s, \upsilon, p \rangle$, such that for some clock $x \in act(s)$, $\upsilon(x) = -\varepsilon < 0$ and $\upsilon(y) \leq \upsilon(x)$ for all $y \in act(s)$, then, for all $0 < \varepsilon' \leq \varepsilon$, $Y^*_{t+\varepsilon'} = \langle s, \upsilon + \varepsilon', p' \rangle$, where $p' = \{\langle q, \nu' \rangle \mid \text{there is } \langle q, \nu \rangle \in p \text{ such that } \nu' = \nu + \varepsilon'\}$;

- *Transition states*: consider Y^* in a state $\langle s, \upsilon, p \rangle$, such that $\upsilon(x) = 0$ for some $x \in act(s)$. Let $a = \{x \mid \upsilon(x) = 0\}$. The process moves to some state $\langle s', \upsilon', p' \rangle$, where

- $next(s, a, s') > 0$;
- For all $x \in old(s, a, s')$, $v'(x) = v(x)$. For all $x \in new(s, a, s')$, $v'(x)$ is according to $dist(x)$;
- $p' = \{\langle q', \nu' \rangle \mid$ there is $\langle q, \nu \rangle \in p$ and there is $\langle q, q', a, \lambda, \delta \rangle \in T$ such that ν satisfies δ and $\nu'(x) = 0$ if $x \in \lambda$ and $\nu'(x) = \nu(x)$ otherwise$\}$.

The Markov process Y^* induces the same probability measure as Y over Σ^t. A particular behavior ρ of Y^* is a function from $\mathbb{R}$ to the state space of Y^*, defined according to the above rules. In the next section we will introduce the idea of a *generic clock* and define the traditional (integral parts/order of fractional parts) equivalence relation on the states of Y^*. Later on we will see that for most practical processes $\mathcal{M}$, the equivalence relation allows us to analyze the process Y^* with the very same method of [2] and decide the verification problem for any timed automaton $\mathcal{A}$.

A.4.1 Generic Clocks

Consider Y^* in a time-passage state $\langle s, v, p \rangle$, where $|p| = n$. In the next state $\langle s, v+\varepsilon', p' \rangle$, $|p'|$ is still n. If Y^* is in a transition state, where $|p| = n$, then, in the next state, $|p'|$ can be as high as kn, where k is the degree of nondeterminism of $\mathcal{A}$, that is, the maximum number of transitions, on the same symbol, that can be simultaneously enabled. But note, however, that the number of distinct values in the ranges of the functions ν in all generalized locations of p' is, at most, one more than the number of distinct values in p. This possible additional distinct value is zero, and it corresponds to all the clocks that were reset by the transition.

Let $c_{\mathcal{A}}$ be the greatest constant appearing in the clock constraints of $\mathcal{A}$. Let $\uparrow$ be a special symbol representing any value in the interval $(c_{\mathcal{A}}, \infty)$. By definition, $\uparrow > c_{\mathcal{A}}$. Given a set p of generalized locations of $\mathcal{A}$, we define the set $R_p \subset [0, c_{\mathcal{A}}] \cup \{\uparrow\}$ as follows: let $R'_p = \{d \mid$ there is $\langle q, \nu \rangle \in p$, such that $d = \nu(x) \leq c_{\mathcal{A}}$ for some $x \in X\}$. If there is $\langle q, \nu \rangle \in p$, such that $\nu(x) > c_{\mathcal{A}}$ for some $x \in X$, then $R_p = R'_p \cup \{\uparrow\}$, otherwise $R_p = R'_p$. In order to formalize the equivalence relation on the states of Y^*, we think of each value in R_p as being represented by a *generic* clock. We create a set of generic clock variables, $C_p = \{c_1, c_2, \ldots, c_{|R_p|}\}$ for p. We define also the bijective function $\eta_p : C_p \to R_p$ as the unique function such that $\eta_p(c_1) < \eta_p(c_2) < \cdots < \eta_p(c_{|R_p|})$. Given two sets p and p' of

generalized locations of $\mathcal{A}$, if $|R_p| = |R_{p'}|$, then we interpret the two sets C_p and $C_{p'}$ as being the same set of generic clock variables.

The function η_p induces, for each $\langle q, \nu \rangle \in p$, a function $\mu : X \to C_p$ that associates to each clock $x \in X$ the generic clock which holds "the value $\nu(x)$", that is, $\mu(x) = \eta_p^{-1}(\nu(x))$ if $\nu(x) \leq c_{\mathcal{A}}$ and $\mu(x) = \eta_p^{-1}(\uparrow)$ otherwise. The generalized location $\langle q, \nu \rangle$ is, then, represented by a pair $\langle q, \mu \rangle$, which we call a *position* of $\mathcal{A}$. Note that two different generalized locations can be associated to the same position. This is because all values greater than $c_{\mathcal{A}}$ are mapped to $\uparrow$. For a set of generalized locations p, we define the set of positions of $\mathcal{A}$ as $P_p = \{\langle q, \mu \rangle \mid \langle q, \mu \rangle$ represents some $\langle q, \nu \rangle \in p\}$.

Example 3 Suppose $c_{\mathcal{A}} = 4$ for a timed automaton $\mathcal{A}$ with $Q = \{q_1, q_2, \ldots, q_{10}\}$, and consider the following set of generalized locations of $\mathcal{A}$:

$$p = \left\{ \langle q_2, \begin{bmatrix} x_1(3.1) \\ x_2(4.1) \\ x_3(2.9) \end{bmatrix} \rangle, \langle q_3, \begin{bmatrix} x_1(2.9) \\ x_2(5) \\ x_3(1.3) \end{bmatrix} \rangle, \langle q_8, \begin{bmatrix} x_1(2.9) \\ x_2(\pi) \\ x_3(1.3) \end{bmatrix} \rangle, \langle q_6, \begin{bmatrix} x_1(4.8) \\ x_2(2.2) \\ x_3(2.9) \end{bmatrix} \rangle \right\} ,$$

Then $C_p = \{c_1, c_2, \ldots, c_6\}$,

$$\eta_p = \begin{bmatrix} c_6(\uparrow) \\ c_5(\pi) \\ c_4(3.1) \\ c_3(2.9) \\ c_2(2.2) \\ c_1(1.3) \end{bmatrix} \text{ and } P_p = \left\{ \langle q_2, \begin{bmatrix} x_1(c_4) \\ x_2(c_6) \\ x_3(c_3) \end{bmatrix} \rangle, \langle q_3, \begin{bmatrix} x_1(c_3) \\ x_2(c_6) \\ x_3(c_1) \end{bmatrix} \rangle, \langle q_8, \begin{bmatrix} x_1(c_3) \\ x_2(c_5) \\ x_3(c_1) \end{bmatrix} \rangle, \langle q_6, \begin{bmatrix} x_1(c_5) \\ x_2(c_2) \\ x_3(c_3) \end{bmatrix} \rangle \right\} .$$

□

Given a state $\langle s, \upsilon, p \rangle$ of Y^*, we define the *clock vector* $\xi_{\langle s,\upsilon,p \rangle}$ as the mapping $\xi_{\langle s,\upsilon,p \rangle} : act(s) \cup C_p \to \mathbb{R} \cup \{\uparrow\}$, induced by υ and η_p, that is, for each $x \in act(s)$, $\xi_{\langle s,\upsilon,p \rangle}(x) = \upsilon(x)$, and for each $c \in C_p$, $\xi_{\langle s,\upsilon,p \rangle}(c) = \eta_p(c)$. A generic clock $c \in C_p$ is said to be *irrelevant* to $\xi_{\langle s,\upsilon,p \rangle}$ if $\xi_{\langle s,\upsilon,p \rangle}(c) = \uparrow$, and *relevant* otherwise. Note that at most one generic clock is irrelevant to a clock vector. The set of relevant generic clocks is denoted by C_p^{rel}. We are now ready to define the traditional [4, 2, 17] equivalence relation $\sim$ over the set of all clock vectors.

Given a number $t \in \mathbb{R}$, $\lfloor t \rfloor$ denote the greatest integer smaller than or equal to t, and $\text{fr}(t) = t - \lfloor t \rfloor$ is the fractional part of t. Define $\xi \sim \xi'$ iff:

- *domain*: the domains of ξ and ξ' are equal. Let *act* denote the set of active basic events and let C denote the set of generic clocks in ξ and ξ';

- *irrelevant clock*: for each $x \in C$, $\xi(x) = \uparrow$ iff $\xi'(x) = \uparrow$;
- *exponential basic events*: for each $x \in act \cap E_e$, $\xi(x) = 0$ iff $\xi'(x) = 0$;
- *relevant clocks and bounded basic events*: Let $E^* = (act \cap E_b) \cup C^{\text{rel}}$. (1) for each $x \in E^*$, $\lfloor \xi(x) \rfloor = \lfloor \xi'(x) \rfloor$ and $\text{fr}(\xi(x)) = 0$ iff $\text{fr}(\xi'(x)) = 0$; (2) and for each pair x and y in E^*, $\text{fr}(\xi(x)) < \text{fr}(\xi(y))$ iff $\text{fr}(\xi'(x)) < \text{fr}(\xi'(y))$ and $\text{fr}(\xi(x)) = \text{fr}(\xi(y))$ iff $\text{fr}(\xi'(x)) = \text{fr}(\xi'(y))$.

Finally, we can define the equivalence relation over the set of all states of Y^* as an extension of the relation $\sim$ for clock vectors.

Definition 7 Consider a process $\mathcal{M}$, a timed automaton $\mathcal{A}$, and the associated Markov process Y^*. We define $\langle s, v, p \rangle \sim^* \langle s', v', p' \rangle$ iff:

- $s = s'$;
- $\xi_{\langle s,v,p \rangle} \sim \xi_{\langle s',v',p' \rangle}$;
- $P_p = P_{p'}$.

The relation $\sim^*$ preserves enough information to decide which event $\mathcal{M}$ will deliver next and, when it occurs, which transitions of $\mathcal{A}$ will be enabled. In order to achieve this, the relation $\sim$ records, for each generic clock (condition (1)), the interval from $I^c = \{[0,0], (0,1), [1,1], (1,2), \ldots, [c_{\mathcal{A}}, c_{\mathcal{A}}], (\uparrow)\}$ where the clock is contained. Note that any two clocks in the same interval satisfy the same set of clock constraints. For the basic events, the intervals are from $I^b = \{[-u_g, -u_g], \ldots, (-2,-1), [-1,-1], (-1,0), [0,0]\}$, where u_g is the greatest value in the range of the function u. To correctly update this information, the relation also records (condition (2)) the order of the fractional parts. Nothing is needed, however, for clocks whose values are greater than $c_{\mathcal{A}}$, since all of them satisfy the same set of clock constraints. For exponential events, we need only the intervals $I^e = \{(-\infty, 0), [0,0]\}$, because of the memoryless property of these distributions. Thus, an equivalence class can be specified by a tuple of the form $[s, C, P, int_c, int_b, int_e, Fr]$, where:

- $s \in S$;
- $C = \{c_1, c_2, \ldots, c_K\}$ is a set of generic clocks;

- P is a set of positions of $\mathcal{A}$;
- $int_{\mathrm{c}} : C \to I^{\mathrm{c}}$ gives the interval of each $x \in C$;
- $int_{\mathrm{b}} : act_{\mathrm{b}}(s) \to I^{\mathrm{b}}$ gives the interval of each $x \in act_{\mathrm{b}}(s)$;
- $int_{\mathrm{e}} : act_{\mathrm{e}}(s) \to I^{\mathrm{e}}$ gives the interval of each $x \in act_{\mathrm{e}}(s)$;
- Fr is an ordering for the fractional parts of the clocks in $E^* = act_{\mathrm{b}}(s) \cup C^{\mathrm{rel}}$. It has the form[2] $Fr : 0 \diamond \mathrm{fr}(x_1) \diamond \mathrm{fr}(x_2) \diamond \cdots \diamond \mathrm{fr}(x_{|E^*|}) < 1$, where $\diamond \in \{=, <\}$ and $x_i \in E^*$. Also, int_{c} and Fr are such that $c_1 < c_2 < \cdots < c_K$.

Note 1 Throughout the paper we use the following schematic representation for equivalence classes in $\sim^*$:

$$[s, \begin{bmatrix} c_4(\uparrow) \\ c(-1,0) \\ c_3(2,3) \\ a(-2,-1) \\ c_1(0,1) \\ c_2(1) \\ b(-\infty,0) \end{bmatrix}, P] \ .$$

Each clock x is annotated with its interval (we write $x(c)$ as a shortcut for $x[c,c]$). Between brackets we construct a stack of the relevant generic clocks and the bounded basic events. The stack gives the order of the fractional parts, such that the clock on the top has the greatest fractional part. If there is an irrelevant clock then we put it above the stack. If there are exponential basic events we put them under the stack.

Remark 1 The number of equivalence classes of $\sim^*$ is *not* finite, since there is no bound on the number of generic clocks in the set C. However, it is important to note that the number of equivalence classes with at most K generic clocks *is* finite. Let V_K denote the set of all equivalence classes with exactly K generic clocks, that is, $|C| = K$. The following bound holds (compare to [2, 4]):

$$|V_K| < \underbrace{2^{|Q||C|^{|X|}}}_{(1)} \cdot \underbrace{(2c_{\mathcal{A}}+2)^{|C|}}_{(2)} \cdot \sum_{s \in S} \left[\underbrace{|act_{\mathrm{b}}(s) \cup C|!}_{(3)} \cdot \underbrace{(2u_g+1)^{|act_{\mathrm{b}}(s)|}}_{(4)} \cdot \underbrace{2^{|act_{\mathrm{e}}(s)|}}_{(5)} \right],$$

where the factor (i) refers to the number of possible:

[2] This notation comes from [17].

(1) sets of positions of $\mathcal{A}$;

(2) combinations of intervals for relevant generic clocks;

(3) orders of fractional parts;

(4) combinations of intervals for bounded events;

(5) combinations of intervals for exponential events.

A.4.2 The Graph G

Let V denote the set of all equivalence classes of $\sim^*$. If we project the states of the process Y^* onto their equivalence classes, we get a projected process Y^p over V. By projecting the states of a particular behavior ρ of Y^*, we can obtain an ω-word ρ_p over V, giving the sequence of equivalence classes visited by the behavior ρ.

The process Y^p is not Markov. However, given that Y^p is in a state $v \in V$, we can effectively compute the set of states $\{v' \in V \mid$ there is a positive probability that the next state will be $v'\}$. Thus, we can define an oriented graph G whose vertex set is a subset of V and such that there is an edge vv' iff there is a positive probability that the next state will be v', given that the present state is v. The set V_0 of initial vertices of G is composed by all the vertices of the form $[s, C, P, int_c, int_b, int_e, Fr]$, where:

- $f_0(s) > 0$;
- $C = \{c_1\}$ and $int_c(c_1) = (0)$;
- $P = \{\langle q, \mu\rangle \mid q \in Q_0\}$, where μ is defined as: for each $x \in X$, $\mu(x) = c_1$;
- for every $x \in act_e(s)$, $int_e(x) = (-\infty, 0)$;
- for every fixed-delay basic event $x \in act_b(s)$, $int_b(x) = (-l_x)$;
- for every variable-delay basic event $x \in act_b(s)$, $int_b(x) = (-c, -(c-1))$, where $c \in \mathbb{N}$, $l_x + 1 \leq c \leq u_x$;
- for any two variable-delay basic events x and y in $act_b(s)$, $\mathrm{fr}(x) \neq \mathrm{fr}(y)$.

A state $[s, C, P, int_c, int_b, int_e, Fr]$ of Y^p is called *transient* iff for some $x \in act(s) \cup C^{rel}$, we have $\mathrm{fr}(x) = 0$. The graph G is defined inductively, from the initial vertices, by the following rules that define the edge relation of G.

Rules. The edges are grouped in five different types:

1. Consider Y^p in a transient vertex $[s, C, P, int_c, int_b, int_e, Fr]$, such that for each $x \in act_i(s)$, $int_i(x) \neq (0)$, $i \in \{b, e\}$, and for each $x \in C$, $int_c(x) \neq (c_{\mathcal{A}})$. Then, with probability 1, the next vertex will be $[s, C, P, int'_c, int'_b, int_e, Fr']$, where:

   - for each $x \in C$, $int'_c(x) = (c, c+1)$ if $int_c(x) = (c)$ for some $c \in \mathbb{N}$, and $int'_c(x) = int_c(x)$ otherwise;
   - for each $x \in act_b(s)$, $int'_b(x) = (-c, -(c-1))$ if $int_b(x) = (-c)$ for some $c \in \mathbb{N}$, and $int'_b(x) = int_b(x)$ otherwise;
   - in this case, $Fr : 0 = \mathrm{fr}(x_1) \diamond_2 \mathrm{fr}(x_2) \diamond_3 \cdots \diamond_{|E^*|} \mathrm{fr}(x_{|E^*|}) < 1$. Then, $Fr' : 0 < \mathrm{fr}(x_1) \diamond_2 \mathrm{fr}(x_2) \diamond_3 \cdots \diamond_{|E^*|} \mathrm{fr}(x_{|E^*|}) < 1$.

   For example:

$$[s, \begin{bmatrix} b(-1,0) \\ c_3(2,3) \\ a(-2,-1) \\ c_1(0,1) \\ c_2(1) \end{bmatrix}, P] \xrightarrow{\text{type 1}} [s, \begin{bmatrix} b(-1,0) \\ c_3(2,3) \\ a(-2,-1) \\ c_1(0,1) \\ c_2(1,2) \end{bmatrix}, P] .$$

2. Consider Y^p in a transient vertex $[s, C, P, int_c, int_b, int_e, Fr]$, such that for each $x \in act_i(s)$, $int_i(x) \neq (0)$, $i \in \{b, e\}$, and there is $y \in C$, such that $int_c(y) = (c_{\mathcal{A}})$. Note that there can be only one such y. Then, there are two cases:

   (a) there is no irrelevant generic clock. Then $y = c_{|C|}$, and, with probability 1, the next vertex will be $[s, C, P, int'_c, int'_b, int_e, Fr']$, where:

   - $int'_c(y) = (\uparrow)$, and $int'_c(x) = int_c(x)$ for each $x \in C$, $x \neq y$;
   - for each $x \in act_b(s)$, $int'_b(x) = (-c, -(c-1))$ if $int_b(x) = (-c)$ for some $c \in \mathbb{N}$, and $int'_b(x) = int_b(x)$ otherwise;
   - in this case, $Fr : 0 = \mathrm{fr}(y) \diamond_2 \mathrm{fr}(x_2) \diamond_3 \cdots \diamond_{|E^*|} \mathrm{fr}(x_{|E^*|}) < 1$. Then, $Fr' : 0 < \mathrm{fr}(x_2) \diamond_3 \cdots \diamond_{|E^*|} \mathrm{fr}(x_{|E^*|}) < 1$.

For example, suppose $c_{\mathcal{A}} = 4$:

$$[s, \begin{bmatrix} b(-1,0) \\ c_2(2,3) \\ a(-2,-1) \\ c_1(0,1) \\ c_3(4), c(-1) \end{bmatrix}, P] \xrightarrow{\text{type 2(a)}} [s, \overset{c_3(\uparrow)}{\begin{bmatrix} b(-1,0) \\ c_2(2,3) \\ a(-2,-1) \\ c_1(0,1) \\ c(-1,0) \end{bmatrix}}, P] .$$

(b) there is an irrelevant generic clock. Then $y{=}c_{|C|-1}$, and, with probability 1, the next vertex will be $[s, C', P', int'_{\mathrm{c}}, int'_{\mathrm{b}}, int_{\mathrm{e}}, Fr']$, where:

- $C' = \{c_1, c_2, \ldots, c_{|C|-1}\}$;
- $int'_{\mathrm{c}}(y) = (\uparrow)$, and $int'_{\mathrm{c}}(x) = int_{\mathrm{c}}(x)$ for each $x \in C'$, $x \neq y$;
- for each $x \in act_{\mathrm{b}}(s)$, $int'_{\mathrm{b}}(x) = (-c, -(c-1))$ if $int_{\mathrm{b}}(x) = (-c)$ for some $c \in \mathbb{N}$, and $int'_{\mathrm{b}}(x) = int_{\mathrm{b}}(x)$ otherwise;
- in this case, $|P'| \leq |P|$. $P' = \{\langle q, \mu' \rangle \mid$ there is $\langle q, \mu \rangle \in P$ such that for every $x \in X$, either $\mu'(x) = \mu(x)$ or $\mu'(x) = c_{|C'|}$ and $\mu(x) = c_{|C|}\}$;
- in this case, $Fr : 0 = \mathrm{fr}(y) \diamond_2 \mathrm{fr}(x_2) \diamond_3 \cdots \diamond_{|E^*|} \mathrm{fr}(x_{|E^*|}) < 1$. Then, $Fr' : 0 < \mathrm{fr}(x_2) \diamond_3 \cdots \diamond_{|E^*|} \mathrm{fr}(x_{|E^*|}) < 1$.

For example, let $c_{\mathcal{A}} = 4$ and $X = \{x, y\}$:

$$[s, \overset{c_4(\uparrow)}{\begin{bmatrix} b(-1,0) \\ c_2(2,3) \\ a(-2,-1) \\ c_1(0,1) \\ c_3(4) \end{bmatrix}}, \left\{\langle q_2, \begin{bmatrix} x(c_1) \\ y(c_2) \end{bmatrix}\rangle, \langle q_4, \begin{bmatrix} x(c_4) \\ y(c_1) \end{bmatrix}\rangle, \langle q_4, \begin{bmatrix} x(c_3) \\ y(c_1) \end{bmatrix}\rangle\right\}]$$

$$\downarrow \text{type 2(b)}$$

$$[s, \overset{c_3(\uparrow)}{\begin{bmatrix} b(-1,0) \\ c_2(2,3) \\ a(-2,-1) \\ c_1(0,1) \end{bmatrix}}, \left\{\langle q_2, \begin{bmatrix} x(c_1) \\ y(c_2) \end{bmatrix}\rangle, \langle q_4, \begin{bmatrix} x(c_3) \\ y(c_1) \end{bmatrix}\rangle\right\}] .$$

3. Consider Y^p in a nontransient vertex $[s, C, P, int_{\mathrm{c}}, int_{\mathrm{b}}, int_{\mathrm{e}}, Fr]$, such that $|E^*| \geq 1$. Then, with positive probability, the next vertex will be $[s, C, P, int'_{\mathrm{c}}, int'_{\mathrm{b}}, int_{\mathrm{e}}, Fr']$, where:

- for each $y \in C$, $int'_{\mathrm{c}}(y) = (c+1)$ if $int_{\mathrm{c}}(y) = (c, c+1)$, for some $c \in \mathbb{N}$ and for every $x \in E^*$, $\mathrm{fr}(x) \leq \mathrm{fr}(y)$. Otherwise, $int'_{\mathrm{c}}(y) = int_{\mathrm{c}}(y)$;
- for each $y \in act_{\mathrm{b}}(s)$, $int'_{\mathrm{b}}(y) = (-(c-1))$ if $int_{\mathrm{b}}(y) = (-c, -(c-1))$, for some $c \in \mathbb{N}$ and for every $x \in E^*$, $\mathrm{fr}(x) \leq \mathrm{fr}(y)$. Otherwise, $int'_{\mathrm{b}}(y) = int_{\mathrm{b}}(y)$;
- in this case, $Fr : 0 < \mathrm{fr}(x_1) \diamond_2 \mathrm{fr}(x_2) \diamond_3 \cdots \diamond_M \mathrm{fr}(x_M) < \mathrm{fr}(y_1) = \mathrm{fr}(y_2) = \cdots = \mathrm{fr}(y_N) < 1$. Then, $Fr' : 0 = \mathrm{fr}(y_1) = \mathrm{fr}(y_2) = \cdots = \mathrm{fr}(y_N) < \mathrm{fr}(x_1) \diamond_2 \mathrm{fr}(x_2) \diamond_3 \cdots \diamond_M \mathrm{fr}(x_M) < 1$.

For example:

$$[s, \overset{c_4(\uparrow)}{\begin{bmatrix} b(-1,0), c_2(0,1) \\ c_3(2,3) \\ a(-2,-1) \\ c_1(0,1) \end{bmatrix}}_{c(-\infty,0)}, P] \xrightarrow{\text{type 3}} [s, \overset{c_4(\uparrow)}{\begin{bmatrix} c_3(2,3) \\ a(-2,-1) \\ c_1(0,1) \\ b(0), c_2(1) \end{bmatrix}}_{c(-\infty,0)}, P] .$$

4. Consider Y^p in a nontransient vertex $[s, C, P, int_{\mathrm{c}}, int_{\mathrm{b}}, int_{\mathrm{e}}, Fr]$. Then, for each $y \in act_{\mathrm{e}}(s)$, there is a positive probability that the next vertex will be $[s, C, P, int_{\mathrm{c}}, int_{\mathrm{b}}, int'_{\mathrm{e}}, Fr]$, where:
   - $int'_{\mathrm{e}}(y) = (0)$, and $int'_{\mathrm{e}}(x) = (-\infty, 0)$ for every $x \in act_{\mathrm{e}}(s)$, $x \neq y$;

For example:

$$[s, \overset{c_4(\uparrow)}{\begin{bmatrix} b(-1,0), c_2(0,1) \\ c_3(2,3) \\ a(-2,-1) \\ c_1(0,1) \end{bmatrix}}_{c(-\infty,0)}, P] \xrightarrow{\text{type 4}} [s, \overset{c_4(\uparrow)}{\begin{bmatrix} b(-1,0), c_2(0,1) \\ c_3(2,3) \\ a(-2,-1) \\ c_1(0,1) \end{bmatrix}}_{c(0)}, P] .$$

5. Consider Y^p in a transient vertex $v = [s, C, P, int_{\mathrm{c}}, int_{\mathrm{b}}, int_{\mathrm{e}}, Fr]$ such that $|a| > 0$, $a = \{x \mid x \in act_i(s) \text{ and } int_i(x) = (0),\ i \in \{\mathrm{b}, \mathrm{e}\}\}$. Then, a state transition occurs is $\mathcal{M}$. We need a few definitions.

   Given a position $\langle q, \mu \rangle \in P$, let the clock interpretation ν_μ over X be defined as: $\nu_\mu(x) = (l+u)/2$, where $l = u = c$ if $int_{\mathrm{c}}(\mu(x)) = (c)$, and $l = c$ and $u = c+1$ if

$int_c(\mu(x)) = (c, c+1)$, for some $c \in \mathbb{N}$. We say that μ satisfies a clock constraint δ iff ν_μ satisfies δ.

We say that v is *reseting* iff there is $\langle q, \mu \rangle \in P$ and $\langle q, q', a, \lambda, \delta \rangle \in T$ such that μ satisfies δ and $|\lambda| > 0$. This means that a new generic clock will be needed to represent the value 0 in the next state. Define $D \subseteq \{1, 2, \ldots, |C|\}$ as $D = \{i \mid$ there is $\langle q, \mu \rangle \in P$ such that, for some $x \in X$, $\mu(x) = c_i$ and there is $\langle q, q', a, \lambda, \delta \rangle \in T$ such that $x \notin \lambda$ and μ satisfies $\delta\}$, that is, D contains the indices of all generic clocks whose values will still represent some clock in the next state. Let d be the unique function $d : \{1, 2, \ldots, |D|\} \rightarrow D$ satisfying $d(1) < d(2) < \cdots < d(|D|)$.

Then, there is a positive probability that the next vertex will be any vertex $[s', C', P', int'_c, int'_b, int'_e, Fr']$, where:

- $next(s, a, s') > 0$;
- $C' = \{c_1, c_2, \ldots, c_N\}$, where $N = |D|+1$ if v is reseting and $N = |D|$ otherwise;
- $P' = \{\langle q', \mu' \rangle \mid$ there is $\langle q, \mu \rangle \in P$ and $\langle q, q', a, \lambda, \delta \rangle \in T$ such that μ satisfies δ, and for each $x \in \lambda$, $\mu'(x) = c_1$, and for each $x \notin \lambda$, $\mu'(x) = c_i$, and $\mu(x) = c_{d(i-1)}$ if v is reseting, and $\mu(x) = c_{d(i)}$ otherwise$\}$;
- if v is reseting then $int'_c(c_1) = (0)$, and for $2 \leq i \leq N$, $int'_c(c_i) = int_c(c_{d(i-1)})$. Otherwise, for $1 \leq i \leq N$, $int'_c(c_i) = int_c(c_{d(i)})$;
- for each $x \in old(s, a, s') \cap act_i(s')$, $int'_i(x) = int_i(x)$, $i \in \{b, e\}$;
- for each $x \in new(s, a, s')$:
  - if $x \in act_e(s')$, $int'_e(x) = (-\infty, 0)$;
  - if $x \in act_b(s')$ and is fixed-delay, $int'_b(x) = (-l_x)$;
  - if $x \in act_b(s')$ and is variable-delay, $int'_b(x) = (-c, -(c-1))$, where $c \in \mathbb{N}$, $l_x + 1 \leq c \leq u_x$;
- Let $C^s = C' \setminus \{c_1\}$ if v is reseting, and $C^s = C'$ otherwise. Let $O = (old(s, a, s') \cap act_b(s')) \cup C^s$. Given a clock $x \in O$, if v is reseting, then, $o(x) = c_{d(i-1)}$ if $x = c_i$ and $o(x) = x$ if $x \notin C^s$. Otherwise, $o(x) = c_{d(i)}$ if $x = c_i$ and $o(x) = x$ if $x \notin C^s$.

  Fr' is such that:

- for any two clocks x and y in O, $\mathrm{fr}(x) < \mathrm{fr}(y)$ in Fr' iff $\mathrm{fr}(o(x)) < \mathrm{fr}(o(y))$ in Fr and $\mathrm{fr}(x) = \mathrm{fr}(y)$ in Fr' iff $\mathrm{fr}(o(x)) = \mathrm{fr}(o(y))$ in Fr.
- for any two variable-delay x and y in $new(s, a, s')$, $\mathrm{fr}(x) \neq \mathrm{fr}(y)$, and for any variable-delay $x \in new(s, a, s')$ and $y \in O$, $\mathrm{fr}(x) \neq \mathrm{fr}(y)$.

As an example, consider Y^p in the following state and suppose that there are two edges $\langle q_1, q_2, a, \{x\}, \delta_1\rangle$ and $\langle q_1, q_3, a, \emptyset, \delta_2\rangle$ enabled, that $next(s, a, s') = 1$, and $act(s') = \{b, c\}$, $c \in act_b(s')$ and $l_c = 1$ and $u_c = 2$:

$$[s, \begin{bmatrix} c_2(2,3) \\ b(-3,-2) \\ c_1(0,1) \\ a(0) \end{bmatrix}, \left\{\langle q_1, \begin{bmatrix} x(c_1) \\ y(c_2) \end{bmatrix}\rangle\right\}] .$$

Then Y^p moves, with positive probability, to each state of the form

$$[s', \begin{bmatrix} \longrightarrow \\ c_3(2,3) \\ \longrightarrow \\ b(-3,-2) \\ \longrightarrow \\ c_2(0,1) \\ \longrightarrow \\ c_1(0) \end{bmatrix}, \left\{\langle q_2, \begin{bmatrix} x(c_1) \\ y(c_3) \end{bmatrix}\rangle, \langle q_3, \begin{bmatrix} x(c_2) \\ y(c_3) \end{bmatrix}\rangle\right\}] ,$$

where the arrows indicate the four possible places for $c(-2,-1)$. □

Note that the definition of G implies that, if there is no edge between v and v', then, the probability that the next state will be v' given that the present state is v, is zero. The argument that, indeed, for each edge of G, this probability is positive, also comes from the definition for edges of the types 1, 2 and 5. For the types 3 and 4, the argument is straightforward and is given in the proof of the following lemma in [2].

Lemma 2 (Lemma 2 in [2]) *For each vertex $v \in V$, the set of behaviors ρ of Y^* such that v appears on ρ_p, has positive probability iff there is a finite path in G from some vertex $v_0 \in V_0$ to v.* □

To solve the verification problem, nevertheless, we need to analyze the *ergodic* behavior of Y^*. A strongly connected component $W \subseteq V$ of G is called a *bottom strongly connected component* (b.s.c. component) of G iff for every $v \in W$, if there is an edge vv', then $v' \in W$. The method in [2] relates the ergodic behavior of Y^* to the b.s.c. components of G. Every b.s.c. component must satisfy a certain condition, and the algorithm, in order

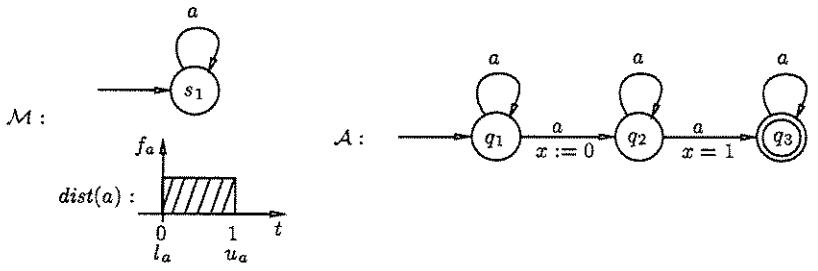

Figura A.3: An instance for which G is infinite

to verify this condition, must visit every vertex of the component. Thus, if G is not finite, the method cannot be applied. We now give an example of an instance for which G is infinite. In the next section, we show that for most interesting practical instances, G *is* finite. Then, in Sect. A.4.4, we will resume the presentation of the algorithm, for these instances for which we can give a guarantee of the finiteness of G.

Example 4 Consider the instance in Fig. A.3. The process $\mathcal{M}$ has a single basic event a with uniform distribution between 0 and 1. Every particular behavior of $\mathcal{M}$ is characterized by a timed word $\rho = (\sigma, \tau)$ such that $\sigma = a^\omega$ and $\tau_i - \tau_{i+1} \leq 1$, $i \geq 0$. The specification $\mathcal{A}$ is the traditional [4] example of a noncomplementable timed automaton. It accepts the language $L(\mathcal{A}) = \{(a^\omega, \tau) \mid$ there are $i \geq 1$ and $j > i$ such that $(\tau_j = \tau_i + 1)\}$. Note that, since $\mathcal{A}$ cannot be complemented and the class of languages defined by deterministic timed automata is closed by complementation, there is no deterministic timed automaton accepting $L(\mathcal{A})$. Clearly, $\mathcal{M}$ does not satisfy $\mathcal{A}$. The probability that a particular behavior of $\mathcal{M}$ satisfies the condition $(\tau_j = \tau_i + 1)$ is zero.

If we try to construct the graph G, we encounter the following infinite sequence of vertices beginning in the unique initial vertex. The label $\delta + a$ in the arrows represents a time passage plus the state transition in $\mathcal{M}$, that is, actually, between any two consecutive vertices in this sequence there are two other vertices corresponding to the passage of time.

$$[s_1, \begin{bmatrix} a(-1,0) \\ c_1(0) \end{bmatrix}, \{\langle q_1, x(c_1)\rangle\}]$$
$$\downarrow \delta + a$$
$$[s_1, \begin{bmatrix} a(-1,0) \\ c_2(0,1) \\ c_1(0) \end{bmatrix}, \{\langle q_1, x(c_2)\rangle, \langle q_2, x(c_1)\rangle\}]$$
$$\downarrow \delta + a$$
$$[s_1, \begin{bmatrix} a(-1,0) \\ c_3(0,1) \\ c_2(0,1) \\ c_1(0) \end{bmatrix}, \{\langle q_1, x(c_3)\rangle, \langle q_2, x(c_1)\rangle, \langle q_2, x(c_2)\rangle\}]$$
$$\downarrow \delta + a$$
$$[s_1, \begin{bmatrix} a(-1,0) \\ c_4(0,1) \\ c_3(0,1) \\ c_2(0,1) \\ c_1(0) \end{bmatrix}, \{\langle q_1, x(c_4)\rangle, \langle q_2, x(c_1)\rangle, \langle q_2, x(c_2)\rangle, \langle q_2, x(c_3)\rangle\}]$$
$$\vdots$$

The number of relevant generic clocks needed to keep track of all the possible runs of $\mathcal{A}$ over this behavior of $\mathcal{M}$ is not bounded, because the clock x is reset, in each run, at different times, so that we need to add one generic clock for each time. In addition, the process can schedule the basic event a arbitrarily close to zero in the interval $(-1, 0)$, such that no generic clock becomes irrelevant. But note that this infinite sequence only exists in G because the process can generate arbitrarily many events in a finite time interval. We show, in the next section, that this is a necessary condition for G to be infinite.

This problem is somehow expected, since the fact that the automaton can cycle arbitrarily many times in q_2, without reseting x, before passing to q_3, is precisely one of the reasons why this automaton is not complementable [4]. □

A.4.3 Instances with Finite G

One may argue that a system model which can generate arbitrarily many discrete events in a finite interval of time is not realistic, since this is not physically realizable. In addition, allowing this property in the model may drastically affect the complexity of the decision problems—it is an essential property in the proofs of the undecidability results about nondeterministic timed automata [4, 27]. Indeed, nondeterministic timed

automata has been recognized as *too* expressive—to the point where the important (for automatic verification) problem of language inclusion is undecidable [5]. There has been much discussion about the adequacy of the various models (see [28] for a start). We will not pursue this discussion here. We show that our use of generic clocks to "encode" the nondeterminism of the automaton suffices to decide the verification problem for processes $\mathcal{M}$ satisfying the following (*K-transitions*) assumption: there is a constant $K \in \mathbb{N}$ such that at most K state transitions can happen, in $\mathcal{M}$, in a time interval of unit length. Thus, one can use the full expressiveness power of nondeterministic timed automata to specify properties as long as the process satisfies this assumption.

This assumption is commonly adopted in an important application of real-time verification techniques: the analysis of digital circuits [34, 9, 35]. For instance, in [9, 35] a timed automaton model for asynchronous circuits is proposed. Every logical gate is followed by a delay element constraining, between lower and upper bounds, the rising and falling (which are the discrete events) of the digital signals. The lower bound is a positive constant, such that any cycle in the model takes at least k time units to complete, for some positive constant k, and so the assumption is met. It is worth noting that the K-transitions assumption does not affect the dense time assumption. In fact, one of the results in [9] is that cyclic circuits in that model, which meets the K-transitions assumption, in general, do not admit discretization.

Lemma 3 *G is infinite iff, for every $k \in \mathbb{N}$, $k > 0$, a vertex with exactly k generic clocks is reachable from some initial vertex.*

Proof 1 The "if" part is trivial. For the other direction, recall that every initial vertex has exactly one generic clock, and that the number of vertices with at most k generic clocks is finite. Thus, if G is infinite, for every $l \in \mathbb{N}$, a vertex with more than l generic clock is reachable. But, by definition, if there is an edge vv', and v has m generic clocks, then v' has, at most, $m+1$ generic clocks. This means that a vertex with n generic clocks is reachable only if a vertex with $n-1$ is also reachable. □

Recall the rule for edges of type 5 in G. Given the transient vertex $v = [s, C, P, int_c, int_b, int_e, Fr]$, we defined a set D containing the indices of the generic clocks in C which represent at least one clock $x \in X$, in some position of $\mathcal{A}$, which is not reset by the transition. We say that a generic clock c_i *survives* the transition if $i \in D$. Its value will

be represented, in the next state, by a generic clock c_j, such that j is, at most, $i+1$ ($j = i+1$ exactly in the case when all generic clocks c_h, $h \leq i$, survive the transition and, in addition, v is reseting). We say that c_j *survived* the transition when it represents some clock $x \in X$, in some position of $\mathcal{A}$, which was not reset by the last transition. Extending this discussion for more than one transition, we see that the index of a generic clock gives a lower bound on the number of transitions that it has survived, and the following lemma holds.

Lemma 4 *Consider a generic clock c_i in any vertex v of G. Then, in any finite initial path of G ending in v, the generic clock c_i survives for, at least, the last $i-1$ transitions.* □

This lemma has also the following interpretation: if a vertex v has k generic clocks, then any finite behavior of $\mathcal{M}$ ending in v has, at least, one run of $\mathcal{A}$ over it, such that some clock $x \in X$ is not reset in this run, for, at least, the last $k-1$ transitions.

Theorem 1 *If $\mathcal{M}$ satisfies the K-transitions assumption, then G is finite.*

Proof 2 Since $\mathcal{M}$ satisfies the K-transitions assumption, we can find a constant N such that any sequence of N transitions in $\mathcal{M}$ takes *more* than $c_{\mathcal{A}}$ time units. Assume that a vertex v with $N+2$ generic clocks is reachable in G from some initial vertex. Then, c_{N+1} is relevant in v, by definition. But this is a contradiction, since Lemma 4 implies that c_{N+1} survives at least the last N transitions, so it cannot be relevant. Since no vertex with $N+2$ generic clocks can be reached, G is finite by Lemma 3. □

Remark 2 The K-transitions assumption is a sufficient but not necessary condition for the finiteness of G. The process may have a cycle where the lower bound of all basic events is zero, but this cycle may not "synchronize" (as it did in Example 4) with a similar cycle in $\mathcal{A}$, where some clock is never reset. Also, clearly, if $\mathcal{A}$ happens to be such that every clock is reset in every cycle, then G is finite for any $\mathcal{M}$.

A.4.4 Ergodic Components and Recurrent Vertices

We now finish the presentation of the algorithm. This final part is a combination of the method in [2] and the results of [16, Section 4]. From now on, assume that G is finite and

let V_G be its set of vertices (V_G is a finite subset of V). Every b.s.c. component of G is classified as either *accepting* or *rejecting*, so that $\mathcal{M}$ satisfies $\mathcal{A}$ iff all b.s.c. components of G are accepting.

Given a particular behavior ρ of Y^*, let $inf(\rho_p) \subseteq V_G$ be the set of vertices of G appearing infinitely often in ρ_p. The main result in [2], which applies directly to our construction, is the following:

Lemma 5 (Lemma 3 in [2]) *Let $W \subseteq V_G$ be a set of vertices of G. Assume that the process Y^* starts in some state $\langle s, v, p\rangle \in v$, for some $v \in W$. The set of behaviors ρ of Y^* such that $inf(\rho_p) = W$ has positive measure iff W is a b.s.c. component of G.* □

We discuss the difficulty behind this lemma. In the discrete time model [16], where the system is given as a Markov chain and the specification as an ω-automaton, the analogous lemma follows trivially from this property: if a vertex v is visited infinitely often by the process, then, each vertex v', such that there is an edge vv', is also visited infinitely often. This is because, if there is an edge from v to v', then the probability that the next vertex will be v' given that the present vertex is v is not only positive, but is a positive constant. In our case, this does not hold. Recall the example for edges of type 5:

$$v = [s, \begin{bmatrix} c_2(2,3) \\ b(-3,-2) \\ c_1(0,1) \\ a(0) \end{bmatrix}, \left\{\langle q_1, \begin{bmatrix} x(c_1) \\ y(c_2) \end{bmatrix}\rangle\right\}]$$

$$\downarrow \text{type 5}$$

$$v' = [s, \begin{bmatrix} c_3(2,3) \\ c(-2,-1) \\ b(-3,-2) \\ c_2(0,1) \\ c_1(0) \end{bmatrix}, \left\{\langle q_2, \begin{bmatrix} x(c_1) \\ y(c_3) \end{bmatrix}\rangle, \langle q_3, \begin{bmatrix} x(c_2) \\ y(c_3) \end{bmatrix}\rangle\right\}] .$$

If the process Y^p is in v, then the probability that the next vertex will be v' is positive, but its exact value depends on the difference $[\mathrm{fr}(c_2) - \mathrm{fr}(b)]$ in v. There is no a priori guarantee that, if the process visits v infinitely often, then it will visit v' also infinitely often, because the difference $[\mathrm{fr}(c_2) - \mathrm{fr}(b)]$ could converge to zero.

Given a vertex $v = [s, C, P, int_c, int_b, int_e, Fr]$, consider the set $E^* = act_b(s) \cup C^{rel}$. We say that a visit of Y^p to v is *δ-separated* iff, for every pair x and y in E^*, if $\mathrm{fr}(x) \neq \mathrm{fr}(y)$ then $|\mathrm{fr}(x) - \mathrm{fr}(y)| \geq \delta$. We will need the following two facts, which are demonstrated in the course of the proof of this Lemma 5:

Fact 1 *Consider a vertex v' such that there is the edge vv'. There exists a positive ε and a positive δ' such that, for any δ-separated visit of Y^p to v, the probability that the next state will be a δ'-separated visit to v', is bounded from below by ε.*

Fact 2 *Given any particular behavior ρ of Y^*, if $inf(\rho_p) = W$, then for each $v \in W$, there is a positive δ_v, such that ρ makes infinitely many δ_v-separated visits to v.*

Lemma 5 gives the relationship between the ergodic behavior of Y^* and the b.s.c. components of G. In order to classify these components, we need some definitions. A vertex $v \in V$, $v = [s, C, P, int_c, int_b, int_e, Fr]$, is called a *unit* vertex, iff $|P| = 1$. A unit vertex $[s, C, \{\langle q, \mu\rangle\}, int_c, int_b, int_e, Fr]$ is said to be *accepting* iff $q \in F$. We say that a unit vertex v is *contained* in a vertex $v' \in V$ iff:

- Given two states of Y^*, $\langle s, v, p\rangle \in v$ and $\langle s', v', p'\rangle \in v'$, there is $\langle q', \nu'\rangle \in p'$ such that, $\langle s, v, p\rangle \sim^* \langle s', v', \{\langle q', \nu'\rangle\}\rangle$. That is, informally, there is a position in v' such that we can obtain v from v' by deleting all the other positions (and modifying the set of generic clocks accordingly).

For example, v is contained in v':

$$v = [s, \begin{bmatrix} c_2(2,3) \\ c(-2,-1) \\ b(-3,-2) \\ c_1(0,1) \end{bmatrix}, \left\{\langle q_3, \begin{bmatrix} x(c_1) \\ y(c_2) \end{bmatrix}\rangle\right\}]$$

$$\cap$$

$$v' = [s, \begin{bmatrix} c_3(2,3) \\ c(-2,-1) \\ b(-3,-2) \\ c_2(0,1) \\ c_1(0) \end{bmatrix}, \left\{\langle q_2, \begin{bmatrix} x(c_1) \\ y(c_3) \end{bmatrix}\rangle, \langle q_3, \begin{bmatrix} x(c_2) \\ y(c_3) \end{bmatrix}\rangle\right\}] .$$

Given a unit vertex $v \in V$, let G_v denote the graph obtained inductively from v by applying the same rules for the edge relation of G. It is important to note that, if v is contained in some vertex of G, then G_v is guaranteed to be finite.

Definition 8 A unit vertex $v \in V$ is called *recurrent* iff, the graph G_v has a b.s.c. component M such that there is a vertex $v' \in M$, and v is contained in v'. We associate with a given recurrent unit vertex v, a finite path γ_v of G_v, going from v to some vertex in M (any such path).

Finally, a b.s.c. component W of G is accepting iff some vertex $v' \in W$ contains an accepting recurrent unit vertex v. Otherwise, it is rejecting.

Lemma 6 (Analogous to Lemma 4.1.2. in [16]) *Let $W \subseteq V_G$ be an accepting b.s.c. component of G. Any behavior ρ of Y^* such that $inf(\rho_p) = W$ is accepted by $\mathcal{A}$ with probability one.*

Proof 3 (Outline) Given a particular behavior ρ' of Y^*, consider the sequence $\rho'_p = v_1 v_2 \cdots$ of vertices of G traversed by ρ'. We construct a run of $\mathcal{A}$ for ρ' by fixing finite segments of the run, repeating the following procedure: choose an index i and a position $\langle q, \mu \rangle$ in v_i. By the definition of Y^*, any finite behavior of Y^* traversing $v_1 v_2 \cdots v_i$ has an initial finite run r of $\mathcal{A}$ ending in a generalized location $\langle q, \nu \rangle$, for some ν. We fix this finite run by deleting all the other positions in v_i, obtaining a unit vertex u, and letting the process Y^* continues from the vertex u. That is, the remaining sequence $v_i v_{i+1} v_{i+2} \cdots$ is *projected* on the graph G_u, yielding a new sequence $\rho''_p = u u_2 u_3 \cdots$ of vertices of G_u.

As W is accepting, there is a vertex $v' \in W$ containing an accepting recurrent unit vertex $v = [s, C, \{\langle q, \mu \rangle\}, int_c, int_b, int_e, Fr]$, such that G_v has a b.s.c. component M, and there is a vertex $v'' \in M$, and v is contained in v''. The idea is to construct an accepting run for the behavior ρ repeating the location q infinitely many times.

Consider the finite path $\gamma_v = v v_2 v_3 \cdots v_{|\gamma_v|}$ in G_v (from Definition 8). If Y^* is in a δ-separated visit to v, then, by applying Fact 1 repeatedly, the probability that the process Y^* follows the path γ_v is bounded from below by a positive constant ε. Also, as v is contained in v', a δ-separated visit to v' can be viewed as a δ-separated visit to v.

Now, consider the sequence $\rho_p = v_1 v_2 \cdots$. By Fact 2, for infinitely many $i \geq 1$, the visit of the behavior ρ to v_i is a $\delta_{v'}$-separated visit to v'. Since these are also $\delta_{v'}$-separated visits to v, then, with probability one, there is a finite index n, such that $v_n = v'$, and, if we project the sequence $v_n v_{n+1} \cdots$ on G_v, the process Y^* follows the path γ_v and is absorbed by the b.s.c. component M. Thus, we fix the first finite segment of the accepting run by choosing v_n and deleting positions in v' so as to obtain v.

The remaining sequence $v_n v_{n+1} \cdots$ is projected on G_v, and the new sequence is $\gamma_v \cdots$, that is, $v v_2 v_3 \cdots v_{|\gamma_v|} \cdots$. But now, we can apply Lemma 5 and Fact 2 on the graph G_v (with v'' playing the role of v') and repeat the procedure infinitely many times. Since every time the procedure is repeated, we can fix the finite segment of the run with probability

one, the whole run is constructed with probability one. □

Lemma 7 (Analogous to Lemma 4.1.3. in [16]) *Let $W \subseteq V_G$ be a rejecting b.s.c. component of G. Any behavior ρ of Y^* such that $inf(\rho_p) = W$ is rejected by $\mathcal{A}$ with probability one.*

Proof 4 (Outline) If no vertex $v' \in W$ contains an accepting unit vertex v, then by Lemma 5, clearly, ρ is rejected by $\mathcal{A}$ with probability one. Suppose there is an accepting unit vertex $v = [s, C, \{\langle q, \mu \rangle\}, int_c, int_b, int_e, Fr]$ contained in some vertex v' of W. The idea is to show that, for every such v, the probability that $\mathcal{A}$ has a run over ρ repeating q infinitely often is zero.

By applying Lemma 5 to the graph G_v, v is contained in the vertex v_i of any sequence $vv_1v_2\cdots$ of G_v, for only finitely many $i \geq 1$, since any such sequence is eventually absorbed by some b.s.c. component of G_v and v is *not* recurrent. Hence, the probability that a behavior of Y^*, traversing the sequence $vv_1v_2\cdots$ in G_v, has a run of $\mathcal{A}$ repeating q infinitely often is zero.

Now, consider the sequence $\rho_p = v_1v_2\cdots$. By Lemma 5, for infinitely many $i \geq 1$, $v_i = v'$. But, for *any* such v_i, if we fix the first finite segment of the run by choosing v_i, then the probability that a run with this initial segment repeats q infinitely often is zero by the last paragraph. □

Putting everything together we have the following theorem, giving the algorithm:

Theorem 2 *Given a real-time probabilistic process $\mathcal{M}$ and a timed automaton $\mathcal{A}$, such that the graph G is finite, $\mathcal{M}$ satisfies $\mathcal{A}$ iff, for every b.s.c. component W of G, there is at least one vertex $v' \in W$, such that v' contains a recurrent accepting unit vertex v.* □

A.5 Conclusions

In this paper, we presented an extension to the method in [2], for the verification of deterministic timed automata specifications of real-time probabilistic processes, giving a partial answer to the question about the verification of nondeterministic timed automata specifications. Our construction gives a semi-decision procedure, which is guaranteed to finish, for instance, when there is a bound on the number of events that the process can

generated in a finite interval of time. This assumption is frequently adopted in practical applications of verifications techniques [34, 9, 35]. Hence, the very expressive formalism of nondeterministic timed automata can be used to specify properties for these probabilistic processes.

The method *is* quite expensive, as can be seen in the bound on the number of equivalence classes with K generic clocks in Section A.4.1. In particular, it is doubly exponential in the number of clocks of the automaton, and exponential in the number of locations of the automaton. Nevertheless, it has been shown, in similar contexts [18, 23], that heuristic methods and symbolic techniques can sometimes yield useful implementations for such expensive (or undecidable, for that matter) problems. Thus, one direction for future work is the development of such methods and techniques for the probabilistic verification problem.

Another direction is, of course, the theoretical question of the decidability of the general verification problem, which remains open. From this theoretical point of view, it is interesting to note that, given an arbitrary process $\mathcal{M}$ and an arbitrary automaton $\mathcal{A}$, such that the procedure does not finish, the present method can be used to decide the verification problem for a modified process $\mathcal{M}_\varepsilon$, which can be arbitrarily "close" to $\mathcal{M}$. First note that the exact form of the probability density functions of the basic events is not relevant, since we are doing qualitative probabilistic verification. Given any rational constant ε, we obtain $\mathcal{M}_\varepsilon$ from $\mathcal{M}$ by:

- replacing the lower bound l_x of every bounded basic event x, such that $l_x = 0$, by ε;
- shifting every exponential distribution to the right by ε. The algorithm can be easily modified to deal with shifted exponential distributions.

Now, we use the transformation of section A.3.2 to get only integer constants; and then use the instance composed of $\mathcal{M}_\varepsilon$ and $\mathcal{A}$, for which the procedure is guaranteed to finish. Moreover, the behavior of $\mathcal{M}_\varepsilon$ approaches the behavior of $\mathcal{M}$ when ε tends to zero; and yet, for any $\varepsilon > 0$, the problem is decidable.

Apêndice B

Autômatos Temporizados e Inclusão de Linguagens

A Semi-decision Procedure for Testing Language Inclusion of Nondeterministic Timed Automata

Arnaldo V. Moura and Guilherme A. Pinto
{arnaldo,guialbu}@dcc.unicamp.br

Abstract

We give a new semi-decision procedure for testing language inclusion of nondeterministic timed automata (NTA). Using this procedure, we show that the language generated by a *progressive* timed automaton can be tested for inclusion against the language generated by *any* NTA. This adds to the known decision procedures concerning NTA. In practice, many timed automata models of actual physical systems are progressive, including models of asynchronous digital circuits. This allows us to retain the full expressiveness of NTA to specify real-time properties. Interestingly enough, the semi-decision procedure is also a reduction of the language inclusion problem for NTA to the language inclusion problem for nondeterministic effective infinite-state ω-automata.

B.1 Introduction

Timed automata (TA) were proposed in [4] as a formalism for the verification of real-time systems. The formalism has been extensively studied and applied to practical problems (see [13] for various references). In the general verification problem, the system and the specification (the desired property) are modeled as TA, so that the problem reduces to testing language inclusion, which is undecidable for nondeterministic timed automata (NTA) [4]. One solution, frequently proposed in the literature, is to use, for the specification, a less expressive formalism, in such a way that the problem becomes decidable. Two such formalisms are deterministic TA [4] and event-clock TA [5]. On the other hand, the notion of nondeterminism facilitates the specification of properties and gives rise to, potentially, smaller models. For these reasons, the investigation of more powerful decision procedures for NTA is a problem of considerable interest.

In this paper, we give a new semi-decision procedure for testing language inclusion of arbitrary NTA. The procedure generalizes the region graph [4] used to solve the emptiness problem. It consists of a subset construction over a parallel composition of the two automata. The composition is guided by the system model automaton, and the two automata synchronize through a set of common *generic* clocks. The undecidability manifests itself in the fact that the system and the specification may synchronize in such a way that an unbounded number of generic clocks is needed. However, we can show that the language generated by a *progressive* TA can be tested for inclusion against the language generated by *any* NTA. In practice, many TA models of actual physical systems are progressive, so that the full expressiveness of NTA can be used to specify real-time properties. These include models of asynchronous digital circuits [9, 35].

In [51], a monadic second-order logic $\mathcal{L}\overleftrightarrow{d}$ is defined and shown to be equally expressive to TA. It is shown that progressive formulas of $\mathcal{L}\overleftrightarrow{d}$ are effectively closed under complementation, establishing that validity is decidable in this case. This gives a decision procedure for the universality problem of progressive timed automata, since [51] also gives effective translations between such formulas and timed automata. Our result concerning progressive automata, Section B.4, gives a decision procedure for testing validity of $\mathcal{L}\overleftrightarrow{d}$ formulas of the type $\phi_1 \Rightarrow \phi_2$, for progressive ϕ_1 and any ϕ_2.

In addition, the semi-decision procedure presented here can also be viewed as a reduction of the language inclusion problem for NTA to the language inclusion problem for

nondeterministic effective infinite-state ω-automata [50]. This can be used to show that the language inclusion problem and the universality problem for NTA are in Π^1_2.

The paper is organized as follows. In Section B.2 we review the formalism of TA. Section B.3 presents the generalization of the region graph, and gives an example for which the semi-decision procedure does not terminate. In Section B.4 we consider the progress condition under which the procedure will always terminate. Section B.5 discusses the problem reduction to infinite-state ω-automata, and Section B.6 concludes with some final remarks.

B.2 Timed Automata

Informally, a timed automaton is a finite-state ω-automaton (see Section B.4) together with a finite set of clock variables whose values increase with the passage of time. Every transition of the automaton has a constraint on the values of the clocks and they can be taken only if the clocks satisfy the constraint. In addition, a transition may reset some of the clocks. TA accept timed words instead of ω-words. A *timed word* ρ, over a finite alphabet of symbols Σ, is a pair (σ, τ) where: $\sigma = \sigma_1\sigma_2\cdots$ is a sequence of symbols $\sigma_i \in \Sigma$ (an ω-word over Σ); and $\tau = \tau_1\tau_2\cdots$ is an strictly increasing sequence of time values $\tau_i \in \mathbb{R}$ (the set of non-negative real numbers), $\tau_i > 0$, satisfying the *progress* property: for every $t \in \mathbb{R}$, there is some $i \geq 1$ such that $\tau_i > t$. In a timed word (σ, τ), the time value τ_i is interpreted as the time when event σ_i occurs. Given a finite set X of clock variables, a *clock constraint* δ over X is defined inductively by $\delta := x \leq c \,|\, c \leq x \,|\, \neg\delta \,|\, \delta_1 \wedge \delta_2$, where $x \in X$ and $c \in \mathbb{Q}$ (the set of non-negative rational numbers). The set of all clock constraints over X is denoted by $\Phi(X)$.

A *timed Büchi automaton* $\mathcal{A}$ is a tuple $\langle \Sigma, Q, Q_0, X, T, F \rangle$, where

- Σ is a finite alphabet of symbols;
- Q is a finite set of locations;
- $Q_0 \subseteq Q$ is a set of start locations;
- X is a finite set of clocks;

- $T \subseteq Q \times Q \times \Sigma \times 2^X \times \Phi(X)$ is a set of transitions. For a transition $\langle q, q', a, \lambda, \delta \rangle$ from location q to location q', on symbol a, δ gives the constraint to be satisfied and λ gives the set of clocks to be reset;

- $F \subseteq Q$ is a set of accepting locations.

The language accepted by $\mathcal{A}$ is obtained by defining runs of $\mathcal{A}$ over timed words. For this, let a *clock interpretation for* X be a function from X to $\mathbb{R}$, that is, a particular reading of the clocks in X. A *generalized location* of $\mathcal{A}$ has the form $\langle q, \nu \rangle$, where $q \in Q$ and ν is a clock interpretation for X. For $t \in \mathbb{R}$, we write $\nu + t$ for the clock interpretation which maps every clock x to $\nu(x) + t$. A clock interpretation ν for X *satisfies* a clock constraint δ over X iff δ evaluates to true when each clock x is replaced by $\nu(x)$.

A *run* r of $\mathcal{A}$ over a timed word $\rho = (\sigma, \tau)$, is a pair $(\overline{q}, \overline{\nu})$ where: $\overline{q} = q_1 q_2 \cdots$ is a sequence of locations in Q; and $\overline{\nu} = \nu_1 \nu_2 \cdots$ is a sequence of clock interpretations for X; satisfying:

- *Initiation*: $q_0 \in Q_0$, and $\nu_0(x) = 0$ for all $x \in X$;

- *Consecution*: for all $i \geq 1$, there exists $\langle q_{i-1}, q_i, \sigma_i, \lambda_i, \delta_i \rangle \in T$ such that $(\nu_{i-1} + \tau_i - \tau_{i-1})$ satisfies δ_i, and $\nu_i(x) = 0$ if $x \in \lambda_i$ and $\nu_i(x) = \nu_{i-1} + \tau_i - \tau_{i-1}$ otherwise ($\tau_0 = 0$, by definition).

Given a run $r = (\overline{q}, \overline{\nu})$ over a timed word ρ, let $inf(r)$ be the set of locations of $\mathcal{A}$ such that $q \in inf(r)$ iff $q = q_i$ for infinitely many $i \geq 1$. The run r over ρ is called an *accepting* run iff $inf(r) \cap F \neq \emptyset$. Finally, the language accepted by $\mathcal{A}$ is $L(\mathcal{A}) = \{(\sigma, \tau) \mid \mathcal{A} \text{ has an accepting run over } (\sigma, \tau)\}$.

One natural way to define the verification problem is to model both the system and the specification (the desired property) as TA. Throughout the paper, $\mathcal{A}_1$ and $\mathcal{A}_2$ always denote the TA giving the system and the specification, respectively. The system satisfies the specification iff $L(\mathcal{A}_1) \subseteq L(\mathcal{A}_2)$. For deterministic[1] $\mathcal{A}_2$, the language inclusion problem reduces to testing emptiness of $L(\mathcal{A}_1) \cap \overline{L(\mathcal{A}_2)}$, which is decidable [4].

[1] A timed automaton is said to be deterministic iff (1) $|Q_0| = 1$, and (2) given any two transitions $\langle q_1, q_1', a_1, \lambda_1, \delta_1 \rangle$ and $\langle q_2, q_2', a_2, \lambda_2, \delta_2 \rangle$ in T, if $q_1 = q_2$ and $a_1 = a_2$, then $\delta_1 \wedge \delta_2$ is unsatisfiable. The interesting property of every deterministic timed automaton is that they have at most one run over every timed word.

The emptiness problem for a TA $\mathcal{A}$ reduces to searching for a special cycle in a so called region graph, which is constructed from an equivalence relation on the set of generalized locations of $\mathcal{A}$ [4]. In the next section, we define a generalization of this region graph, which can be used, in many cases, to decide the language inclusion problem for NTA.

B.3 The Subset Construction Region Graph

Let $\mathcal{A}_1 = \langle \Sigma, Q_1, Q_{01}, X_1, T_1, F_1 \rangle$ and $\mathcal{A}_2 = \langle \Sigma, Q_2, Q_{02}, X_2, T_2, F_2 \rangle$. As in [4], we assume, without loss of generality, that all the constants in all the clock constraints of $\mathcal{A}_1$ and $\mathcal{A}_2$ are integers. We also assume that $\mathcal{A}_1$ and $\mathcal{A}_2$ are disjoint, except for the alphabet Σ. Since we cannot complement $\mathcal{A}_2$ in general [4], in order to cope with the nondeterminism we use the standard idea of a subset construction, applied on a parallel composition of the generalized locations of $\mathcal{A}_1$ and $\mathcal{A}_2$. We will not formally define the parallel composition or the subset construction. These concepts will be implicitly used in the definition of a graph G over which the semi-decision procedure is obtained. The composition is guided by $\mathcal{A}_1$, that is, we take care of only the timed words which have some run of $\mathcal{A}_1$ over it. This is because $L(\mathcal{A}_1) \subseteq L(\mathcal{A}_2)$ iff every timed word that has an accepting run of $\mathcal{A}_1$ over it, also has an accepting run of $\mathcal{A}_2$ over it.

Let A_1 and A_2 denote, respectively, the set of all generalized locations of $\mathcal{A}_1$ and $\mathcal{A}_2$. The basic mathematical object used, from now on, is what we call a *composite pair* for A_1 and A_2, which has the form $\langle p, s \rangle$, where p is a finite subset of A_1, and s is a finite subset of A_2. We denote by $\mathcal{P}$ the set of all composite pairs for A_1 and A_2.

B.3.1 Generic Clocks and the Equivalence Relation

The set $\mathcal{P}$ is uncountable. As in [4], we define an equivalence relation $\sim$ over $\mathcal{P}$ from which we obtain the graph G. It will turn out that the number of equivalence classes in $\sim$ is countable, but it is *not* finite. However, for many interesting instances of the language inclusion problem, G will be finite, a fact that will guarantee termination of the procedure. In order to define the equivalence relation, we need to introduce the idea of a generic clock. We start with the following discussion.

Consider a finite timed word $\varrho = (\sigma_1, \tau_1) \rightarrow (\sigma_2, \tau_2) \rightarrow \cdots \rightarrow (\sigma_i, \tau_i)$. Let $\langle p_\varrho, s_\varrho \rangle$ be a composite pair, where $p_\varrho = \{\langle q, \nu \rangle \mid$ there is a finite run $\langle q_0, \nu_0 \rangle \rightarrow \langle q_1, \nu_1 \rangle \rightarrow \langle q_2, \nu_2 \rangle \rightarrow$

$\cdots \rightarrow \langle q, \nu \rangle$ of $\mathcal{A}_1$ over $\varrho\}$; and let s_ϱ be defined in the same way for $\mathcal{A}_2$. Also, let $D_{\langle p_\varrho, s_\varrho \rangle} = \{t \in \mathbb{R} \,|\, t$ is in the range of some clock interpretation in p_ϱ or in $s_\varrho\}$. The composite pair $\langle p_\varrho, s_\varrho \rangle$ records enough information to determine the future behavior of $\mathcal{A}_1$ and $\mathcal{A}_2$ over any timed word having ϱ as a prefix. Now consider another finite timed word $\varrho' = (\sigma'_1, \tau'_1) \rightarrow (\sigma'_2, \tau'_2) \rightarrow \cdots \rightarrow (\sigma'_{i+1}, \tau'_{i+1})$, such that ϱ' has ϱ as a prefix. Then, $|p_{\varrho'}|$ can be as high as k_1 times $|p_\varrho|$, where k_1 is the degree of nondeterminism[2] of $\mathcal{A}_1$. The same is true for $|s_{\varrho'}|$, k_2 and $|s_\varrho|$. It is interesting to note, however, that $|D_{\langle p_{\varrho'}, s_{\varrho'} \rangle}|$ is, at most, $|D_{\langle p_\varrho, s_\varrho \rangle}| + 1$.

Let α denote the greatest constant appearing in the clock constraints of $\mathcal{A}_1$ and $\mathcal{A}_2$. A value $t \in \mathbb{R}$ is called *relevant* if $t \leq \alpha$, and *irrelevant* otherwise. The above discussion motivates the definition of a generic clock. Informally, given a composite pair $\langle p, s \rangle$, we interpret each relevant value in $D_{\langle p,s \rangle}$ as being held by a generic clock; and all the irrelevant values in $D_{\langle p,s \rangle}$ as being held by a single generic clock. The traditional equivalence relation over the set of clock interpretations [4] is, instead, applied over the set of generic clock interpretations. Let us formalize these notions.

Generic Clocks.

Let $\uparrow$ be a special symbol representing any value in the interval (α, ∞). By definition, $\uparrow > \alpha$. Given a composite pair $\langle p, s \rangle$, we define the set $R_{\langle p,s \rangle} \subset [0, \alpha] \cup \{\uparrow\}$ as follows: let $R'_{\langle p,s \rangle} = \{d \in \mathbb{R} \,|\, d \in D_{\langle p,s \rangle}$ and d is relevant$\}$. If there is an irrelevant value in $D_{\langle p,s \rangle}$, then $R_{\langle p,s \rangle} = R'_{\langle p,s \rangle} \cup \{\uparrow\}$, otherwise $R_{\langle p,s \rangle} = R'_{\langle p,s \rangle}$. We create a set of generic clock variables, $C_{\langle p,s \rangle} = \{c_1, c_2, \ldots, c_{|R_{\langle p,s \rangle}|}\}$ for $\langle p, s \rangle$. The *generic clock interpretation* $\eta_{\langle p,s \rangle}$ is defined as the unique bijective function $\eta_{\langle p,s \rangle} : C_{\langle p,s \rangle} \rightarrow R_{\langle p,s \rangle}$ satisfying $\eta_{\langle p,s \rangle}(c_1) < \eta_{\langle p,s \rangle}(c_2) < \cdots < \eta_{\langle p,s \rangle}(c_{|R_{\langle p,s \rangle}|})$, that is, the generic clock c_i holds the i-th smaller value in $R_{\langle p,s \rangle}$. A generic clock c_i is said to be *irrelevant* to $\eta_{\langle p,s \rangle}$ if $\eta_{\langle p,s \rangle}(c_i) = \uparrow$, and *relevant* otherwise. Note that at most one generic clock is irrelevant to a clock interpretation.

Given two composite pairs $\langle p, s \rangle$ and $\langle p', s' \rangle$, if $|R_{\langle p,s \rangle}| = |R_{\langle p',s' \rangle}|$, then we interpret the two sets $C_{\langle p,s \rangle}$ and $C_{\langle p',s' \rangle}$ as being the same set of generic clock variables. The function $\eta_{\langle p,s \rangle}$ induces, for each $\langle q, \nu \rangle \in p$, a function $\mu : X_1 \rightarrow C_{\langle p,s \rangle}$ that associates to each clock $x \in X_1$ the generic clock which holds the value $\nu(x)$, that is, $\mu(x) = \eta^{-1}_{\langle p,s \rangle}(\nu(x))$

[2]The degree of nondeterminism of a timed automaton $\langle \Sigma, Q, Q_0, X, T, F \rangle$ is the cardinality of the greatest set $E \subseteq T$ such that all transitions in E originate in the same location, are on the same symbol, and the conjunction of their clock constraints can be satisfied.

if $\nu(x) \leq \alpha$ and $\mu(x) = \eta^{-1}_{\langle p,s\rangle}(\uparrow)$ otherwise. The generalized location $\langle q, \nu\rangle$ is, then, *represented* by a pair $\langle q, \mu\rangle$, which we call a *position* of $\mathcal{A}_1$. Note that two different generalized locations can be associated to the same position. This is because all values greater than α are mapped to $\uparrow$. For a composite pair $\langle p, s\rangle$, we define the set of positions of $\mathcal{A}_1$ as $P_{\langle p,s\rangle} = \{\langle q, \mu\rangle \mid \langle q, \mu\rangle$ represents some $\langle q, \nu\rangle \in p\}$. Similarly, we define the set $S_{\langle p,s\rangle}$ with respect to $\mathcal{A}_2$.

The Equivalence Relation.

Now we define the equivalence relation $\sim$ over the set of composite pairs (compare to [4]). Given a number $t \in \mathbb{R}$, $\lfloor t \rfloor$ denotes the greatest integer smaller than or equal to t, and $\mathrm{fr}(t) = t - \lfloor t \rfloor$ denotes the fractional part of t. Define $\langle p, s\rangle \sim \langle p', s'\rangle$ iff:

- *same set of generic clocks*: $C_{\langle p,s\rangle} = C_{\langle p',s'\rangle}$;
- *same sets of positions*: $P_{\langle p,s\rangle} = P_{\langle p',s'\rangle}$, and $S_{\langle p,s\rangle} = S_{\langle p',s'\rangle}$;
- *equivalent generic clocks interpretations*:
  - *irrelevant clock*: for each $x \in C_{\langle p,s\rangle}$, $\eta_{\langle p,s\rangle}(x) = \uparrow$ iff $\eta_{\langle p',s'\rangle}(x) = \uparrow$;
  - *relevant clocks*:
    - for each $x \in C_{\langle p,s\rangle}$ and relevant to $\eta_{\langle p,s\rangle}$, $\lfloor \eta_{\langle p,s\rangle}(x)\rfloor = \lfloor \eta_{\langle p',s'\rangle}(x)\rfloor$ and $\mathrm{fr}(\eta_{\langle p,s\rangle}(x)) = 0$ iff $\mathrm{fr}(\eta_{\langle p',s'\rangle}(x)) = 0$;
    - for each pair x and y in $C_{\langle p,s\rangle}$, both relevant to $\eta_{\langle p,s\rangle}$, $\mathrm{fr}(\eta_{\langle p,s\rangle}(x)) < \mathrm{fr}(\eta_{\langle p,s\rangle}(y))$ iff $\mathrm{fr}(\eta_{\langle p',s'\rangle}(x)) < \mathrm{fr}(\eta_{\langle p',s'\rangle}(y))$, and $\mathrm{fr}(\eta_{\langle p,s\rangle}(x)) = \mathrm{fr}(\eta_{\langle p,s\rangle}(y))$ iff $\mathrm{fr}(\eta_{\langle p',s'\rangle}(x)) = \mathrm{fr}(\eta_{\langle p',s'\rangle}(y))$.

The relation $\sim$ records, for each generic clock, the interval from $\{[0,0], (0,1), [1,1], (1,2), \ldots, [\alpha,\alpha], (\uparrow)\}$ where the clock is contained. Note that any two clocks in the same interval satisfy the same set of clock constraints. To correctly update this information, the relation also records the order of the fractional parts for the relevant clocks. Nothing is needed, however, for clocks whose values are greater than α, since all of them satisfy the same set of clock constraints. We refer the reader to [4] for a detailed discussion about this equivalence relation. In the sequel, we write $[\langle p, s\rangle]$ for the equivalence class to which $\langle p, s\rangle$ belongs.

We finish this section noting that the number of equivalence classes of $\sim$ is not finite, since there is no bound on the number of generic clocks. However, it is important to note that the number of equivalence classes with at most K generic clocks *is* finite. Let V_K denote the set of all equivalence classes with exactly K generic clocks. The following bound holds (compare to [4]):

$$|V_K| < 2^{|Q_1|K^{|X_1|}} \times 2^{|Q_2|K^{|X_2|}} \times (2\alpha+2)^K \times K! \ .$$

B.3.2 Time Successors and the Graph G

Let $\Pi_{\mathcal{P}}$ denote the set of all equivalence classes of the relation $\sim$ over $\mathcal{P}$. We define now the subset construction region graph $G = (V, E)$. Its vertex set V is a subset of $\Pi_{\mathcal{P}} \times \{1, 2\}$. The reason why we need two copies of each equivalence class will be clear soon. G has a unique initial vertex $\langle v_0, 1\rangle$, and each edge is labelled with one symbol from $\Sigma \cup \{\triangleright\}$. The new symbol $\triangleright$ represents a passage of time. Any edge from a vertex $\langle -, 1\rangle$ goes to a vertex $\langle -, 2\rangle$, and it has label $\triangleright$. Any edge from a vertex $\langle -, 2\rangle$ goes to a vertex $\langle -, 1\rangle$, and it has label a, for some $a \in \Sigma$. Thus, the graph G is bipartite. The edge relation is defined in such a way that the graph is "deterministic", in the following sense: let a *run* of G be an infinite sequence of the form: $\langle v_0, 1\rangle \xrightarrow{\sigma_1} \langle v_1, 2\rangle \xrightarrow{\sigma_2} \langle v_2, 1\rangle \xrightarrow{\sigma_3} \cdots$, such that, for all $i \geq 0$, there is an edge from $\langle v_i, -\rangle$ to $\langle v_{i+1}, -\rangle$ with label σ_{i+1}. Given a timed word $\rho = (\sigma, \tau)$, there exists, at most, one run $\langle v_0, 1\rangle \xrightarrow{\triangleright} \langle v_1, 2\rangle \xrightarrow{\sigma_1} \langle v_2, 1\rangle \xrightarrow{\triangleright} \langle v_3, 2\rangle \xrightarrow{\sigma_2} \langle v_4, 1\rangle \xrightarrow{\triangleright} \cdots$ of G, such that, for every $i \geq 1$, the following holds: $[\langle p_{\varrho_i}, s_{\varrho_i}\rangle] = v_{2i}$, where ϱ_i is the finite timed word $(\sigma_1, \tau_1) \rightarrow (\sigma_2, \tau_2) \rightarrow \cdots \rightarrow (\sigma_i, \tau_i)$. We refer to this run as the run of G over the timed word ρ. The undecidability of the inclusion problem manifests itself in the fact that G may be an infinite graph. In Section B.4 we give some sufficient conditions for G to be finite. Once G is finite, we show how one can obtain two Büchi ω-automata $\mathcal{B}_1$ and $\mathcal{B}_2$ such that $L(\mathcal{B}_1) \subseteq L(\mathcal{B}_2)$ iff $L(\mathcal{A}_1) \subseteq L(\mathcal{A}_2)$. Thus, the problem will be reduced, in this case, to language inclusion of ω-automata.

The graph G is constructed inductively, from the initial vertex, by the definition of the edge relation. As in [4], we use the convenient notion of a time successor of an equivalence class to define the edge relation. In order to obtain an effective computational procedure, instead, one should define a representation for the equivalence classes and define the edge relation directly between the vertices. This can certainly be done, although with the cost of considering many different cases in the definition of the edge relation.

Time Successors.

Let v be an equivalence class. Consider a composite pair $\langle p, s\rangle$ in v. Given $t \in \mathbb{R}$, let $\langle p, s\rangle + t$ denote the composite pair obtained from $\langle p, s\rangle$ by replacing every clock interpretation ν in p or in s, by $\nu + t$. An equivalence class v' is a *time successor* of v iff, given $\langle p, s\rangle$ in v, $v' = [\langle p, s\rangle + t]$, for some $t \in \mathbb{R}$, $t > 0$. Any equivalence class v has finitely many time successors, since the number of generic clocks in any time successor of v is, clearly, smaller than or equal to the number of generic clocks in v. In particular, for any t_1 and t_2, both greater than α, $[\langle p, s\rangle + t_1] = [\langle p, s\rangle + t_2]$, which is an equivalence class with only one irrelevant generic clock.

The graph G.

The graph G has a unique initial vertex $\langle v_0, 1\rangle$, where $v_0 = [\langle p_0, s_0\rangle]$ and $p_0 = \{\langle q, \nu_0\rangle \mid q \in Q_{0_1}\}$, and ν_0 is the clock interpretation which maps each $x \in X_1$ to zero. The same definition applies to s_0 with respect to $\mathcal{A}_2$. Note that, in fact, $\langle p_0, s_0\rangle$ is the unique composite pair in v_0. A vertex $\langle v, 1\rangle$ has an edge with label $\triangleright$ to a vertex $\langle v', 2\rangle$ iff v' is a time successor of v. A vertex $\langle v, 2\rangle$ has an edge with label $a \in \Sigma$ to a vertex $\langle v', 1\rangle$ iff:

- Given a composite pair $\langle p, s\rangle$ in v, the following conditions hold:

  1. Given a generalized location $\langle q, \nu\rangle$ in p, there is a transition $\langle q, -, a, -, \delta\rangle$ in T_1, such that ν satisfies δ; and
  2. $v' = [\langle p', s'\rangle]$, where:
     - $p' = \{\langle q', \nu'\rangle \mid$ there is $\langle q, \nu\rangle$ in p, and $\langle q, q', a, \lambda, \delta\rangle$ in T_1, such that ν satisfies δ, and for each $x \in X_1$, $\nu'(x) = 0$ if $x \in \lambda$, and $\nu'(x) = \nu(x)$ otherwise$\}$;
     - $s' = \{\langle q', \nu'\rangle \mid$ there is $\langle q, \nu\rangle$ in s, and $\langle q, q', a, \lambda, \delta\rangle$ in T_2, such that ν satisfies δ, and for each $x \in X_2$, $\nu'(x) = 0$ if $x \in \lambda$, and $\nu'(x) = \nu(x)$ otherwise$\}$.

Note that there is at most one edge out of a vertex $\langle v, 2\rangle$ for each symbol in Σ, and that there may be a vertex $\langle v, 2\rangle$ such that there is no edge out of $\langle v, 2\rangle$. But, the initial vertex and condition 1. guarantee that for every vertex $\langle v, 1\rangle \in V$, the $\mathcal{A}_1$-component is

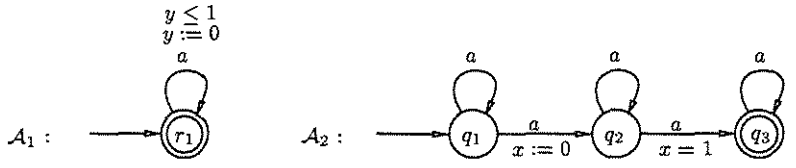

Figura B.1: An instance for which G is infinite

nonempty, that is, p is nonempty for every composite pair $\langle p, s\rangle$ in v. On the other hand, the $\mathcal{A}_2$-component may be empty. We now give an example of an instance for which G is infinite.

Example of Infinite G.

Consider the instance in Fig. B.1. The automaton $\mathcal{A}_2$ is the traditional example of a noncomplementable NTA [4]. Clearly, $L(\mathcal{A}_1) \not\subseteq L(\mathcal{A}_2)$. Consider the following finite timed words $\varrho_i = (a, 1 - 1/2^1) \to (a, 1 - 1/2^2) \to (a, 1 - 1/2^3) \to \cdots \to (a, 1 - 1/2^i)$, $i \geq 1$. The problem occurs because ϱ_i has $i + 1$ different finite runs of $\mathcal{A}_2$ over it, and the clock x is reset at distinct times for each run. Hence $D_{\langle p_{\varrho_i}, s_{\varrho_i}\rangle}$ has $i + 1$ distinct and relevant values, so that $[\langle p_{\varrho_i}, s_{\varrho_i}\rangle]$ has $i + 1$ generic clocks. For any $i \geq 1$, there is a timed word ρ_i, having ϱ_i as a prefix and such that $\mathcal{A}_1$ has a run over ρ_i. Thus, if we try to construct G we will need an infinite sequence of distinct vertices of the form: $\langle v_0, 1\rangle \xrightarrow{\triangleright} \langle -, 2\rangle \xrightarrow{a} \langle [\langle p_{\varrho_1}, s_{\varrho_1}\rangle], 1\rangle \xrightarrow{\triangleright} \langle -, 2\rangle \xrightarrow{a} \langle [\langle p_{\varrho_2}, s_{\varrho_2}\rangle], 1\rangle \xrightarrow{\triangleright} \cdots$.

B.4 Progressive Automata

In the above example, the automaton $\mathcal{A}_1$ has runs over timed words where an arbitrary number of events can happen in a time interval of unit length. Allowing this property in the system model may drastically affect the complexity of the decision problems—it is an essential property in the proofs of the undecidability results about NTA [4, 27]. One may argue that a system model which can generate arbitrarily many discrete events in a finite interval of time is not realistic, since this is not physically realizable. In fact, many TA models of actual physical systems satisfy the property that there is a constant K such that at most K discrete events can happen, in any run, in a time interval of unit length. We say that these TA are *progressive*. For instance, in [9, 35] a TA model for asynchronous circuits

is proposed. Every logical gate is followed by a delay element constraining, between lower and upper bounds, the rising and falling (which are the discrete events) of the digital signals. The lower bound is a positive constant, such that any cycle in the model takes at least k time units to complete, for some positive constant k, and so, the automata are progressive; see also [34, 51] where this same discussion occurs in similar formalisms. It is worth noting that the progress requirement does not nullify the dense time assumption. In fact, one of the results in [9] is that cyclic circuits in that model, in general, do not admit discretization.

We can show that when the system model $\mathcal{A}_1$ is progressive, the graph G is finite for any nondeterministic $\mathcal{A}_2$. Thus, one may use the full expressive power of NTA to specify real-time properties.

Theorem 3 *If $\mathcal{A}_1$ is progressive, then G is finite.*

Proof 5 (Outline) From the definitions, if there is an edge from a vertex $\langle v, 2\rangle$, where v has n generic clocks, to a vertex $\langle v', 1\rangle$, then v' has, at most, $n+1$ generic clocks. Since the number of equivalence classes with at most k generic clocks is finite, then G is infinite iff, for every natural $k > 0$, G has a vertex $\langle v, 1\rangle$, where v has exactly k generic clocks.

Let N be a constant such that for every timed word that has a run of $\mathcal{A}_1$ over it, any sequence of N events takes *more* than α time units. This constant exists since $\mathcal{A}_1$ is progressive. Assume that G has a vertex $\langle v, 1\rangle$, where v has $N+2$ generic clocks. By definition, there are $N+1$ relevant generic clocks in v. We can show, also from the definitions, that the relevant generic clock holding the greatest value in v represents some clock, in some finite run of $\mathcal{A}_1$ or $\mathcal{A}_2$, that is not reset by the last N transitions, for all finite timed words such that the run of G over them ends in $\langle v, 1\rangle$. This is a contradiction to the fact that this generic clock is still relevant. □

We finish this section noting that the progressiveness of $\mathcal{A}_1$ is a sufficient but not necessary condition for the finiteness of G. As an example, consider the instance of Fig. B.1. If we remove the command $x := 0$ from the transition from q_1 to q_2 in $\mathcal{A}_2$, the graph G becomes finite, in spite of the non-progressiveness of $\mathcal{A}_1$. Another sufficient, although hardly acceptable, condition for the finiteness of G is that every clock be reset, at least once, in every cycle of $\mathcal{A}_1$ and $\mathcal{A}_2$.

B.4.1 Obtaining the ω-Automata.

In [4], given a timed automaton $\mathcal{A}$, an ω-automaton $\mathcal{B}$ is obtained from the region graph $R(\mathcal{A})$, such that $L(\mathcal{A}) = \emptyset$ iff $L(\mathcal{B}) = \emptyset$. To this end, a correspondence between the runs of $\mathcal{A}$ and the runs of $R(\mathcal{A})$ is established through the concept of a *progressive* run of $R(\mathcal{A})$. This correspondence readily generalizes to our subset construction.

For a given equivalence class v, we define the set P_v of positions of $\mathcal{A}_1$ as being equal to the set $P_{\langle p,s\rangle}$ for some composite pair $\langle p, s\rangle \in v$ (by definition, the set of positions of $\mathcal{A}_1$ is the same for every composite pair in v). The same definition holds for S_v, with respect to $\mathcal{A}_2$. Consider an edge from a vertex $\langle v, 2\rangle$ to a vertex $\langle v', 1\rangle$ with label a, for some $a \in \Sigma$. This edge naturally induces a relation between the positions in P_v and the positions in $P_{v'}$. We say that a position $\langle q, \mu\rangle$ in P_v is *a-linked* to a position $\langle q', \mu'\rangle$ in $P_{v'}$ iff:

- $\langle q, \mu\rangle$ represents some generalized location $\langle q, \nu\rangle$, and $\langle q', \mu'\rangle$ represents some generalized location $\langle q', \nu'\rangle$; and there is a transition $\langle q, q', a, \lambda, \delta\rangle$ in T_1, such that: ν satisfies δ, and for each $x \in X_1$, $\nu'(x) = 0$ if $x \in \lambda$, and $\nu'(x) = \nu(x)$ otherwise.

Also, consider an edge from a vertex $\langle v, 1\rangle$ to a vertex $\langle v', 2\rangle$ with label $\triangleright$. We say that a position $\langle q, \mu\rangle$ in P_v is *$\triangleright$-linked* to a position $\langle q', \mu'\rangle$ in $P_{v'}$ iff:

- $\langle q, \mu\rangle$ represents some generalized location $\langle q, \nu\rangle$, and $\langle q', \mu'\rangle$ represents some generalized location $\langle q', \nu'\rangle$; and, for some $t \in \mathbb{R}$, $t > 0$, $\langle q', \nu'\rangle = \langle q, \nu\rangle + t$.

Analogously, we can define the "linked" relation between the positions in S_v and S'_v, with respect to $\mathcal{A}_2$. Given a run $r = \langle v_0, 1\rangle \xrightarrow{\sigma_1} \langle v_1, 2\rangle \xrightarrow{\sigma_2} \langle v_2, 1\rangle \xrightarrow{\sigma_3} \cdots$ of G, let an *$\mathcal{A}_1$-run* of r be an infinite sequence $m_0 m_1 m_2 \cdots$ of positions of $\mathcal{A}_1$, such that, for all $i \geq 0$, $m_i \in P_{v_i}$, and m_i is σ_{i+1}-linked to m_{i+1}. In a position $m = \langle q, \mu\rangle$, the function μ maps each clock x in X_1 to a generic clock c_x which is contained in exactly one interval from $\{[0,0], (0,1), [1,1], (1,2), \ldots, [\alpha,\alpha], (\uparrow)\}$. We already known, by the definition of G, that every timed word that has some run of $\mathcal{A}_1$ over it, has a run of G over it. In a timed word ρ, time diverges, so that if r is the run of G over ρ, then, in every $\mathcal{A}_1$-run of r, every clock x in X_1 is either reset (mapped to $[0,0]$) infinitely often, or, after some time, it increases without bound (is continuously mapped to $(\uparrow)$) [4]. Such $\mathcal{A}_1$-runs are called *progressive*. Call a run of G progressive iff it has a progressive $\mathcal{A}_1$-run. The correspondence states

that, for every progressive run r of G, we can find a timed word ρ such that r is the run of G over ρ [4]. Then, clearly, either all $\mathcal{A}_1$-runs of r are progressive, or none is progressive.

We treat first the case where $\mathcal{A}_1$ is known, in advance, to be progressive. In this case, all runs of G are progressive. Afterwards, we discuss how to treat any $\mathcal{A}_1$. Assume that $\mathcal{A}_1$ is progressive. Given a run r of G, consider the set R_r of timed words $R_r = \{\rho \mid r \text{ is the run of } G \text{ over } \rho\}$. We have $|R_r| \geq 1$ (by the above discussion). We say that a position $\langle q, \mu \rangle$ is $\mathcal{A}_1$-*accepting* iff $q \in F_1$. A run $r = \langle v_0, 1\rangle \xrightarrow{\sigma_1} \langle v_1, 2\rangle \xrightarrow{\sigma_2} \langle v_2, 1\rangle \xrightarrow{\sigma_3} \cdots$ of G is $\mathcal{A}_1$-*accepting* iff there is an $\mathcal{A}_1$-run $m_0 m_1 m_2 \cdots$ of r, such that for infinitely many $i \geq 0$, m_i is $\mathcal{A}_1$-*accepting*. The same definitions hold with respect to $\mathcal{A}_2$. Then, either all timed words in R_r are accepted by $\mathcal{A}_1$ or all timed words in R_r are rejected by $\mathcal{A}_1$. The same holds for $\mathcal{A}_2$. Finally, the language inclusion problem reduces to verifying that, for every run r of G, if r is $\mathcal{A}_1$-accepting, then r is $\mathcal{A}_2$-accepting.

We need to cope with a known difficulty of applying a subset construction to a Büchi automaton. Consider a run $r = \langle v_0, 1\rangle \xrightarrow{\sigma_1} \langle v_1, 2\rangle \xrightarrow{\sigma_2} \langle v_2, 1\rangle \xrightarrow{\sigma_3} \cdots$ of G. We *cannot* say that r is $\mathcal{A}_1$-accepting if, for infinitely many $i \geq 0$, there is a $\mathcal{A}_1$-accepting position in P_{v_i}. We refer the reader to [47] for a solution to this difficulty, in the context of ω-automata. As we will see, instead of trying to solve this directly on G, we will consider a nondeterministic image[3] of G (somehow undoing the subset construction), so that the obtained ω-automata will be nondeterministic and the mentioned difficulty is deferred to the algorithms for the language inclusion problem for nondeterministic ω-automata.

ω-Automata.

A *Büchi ω-automaton* $\mathcal{B}$ is a tuple $\langle \Delta, Q, Q_0, T, F\rangle$, where

- Δ is a finite alphabet of symbols;
- Q is a finite set of states;
- $Q_0 \subseteq Q$ is a set of initial states;
- $T \subseteq Q \times Q \times \Delta$ is a set of transitions;
- $F \subseteq Q$ is a set of final states.

[3] This is inspired by the deterministic image of a nondeterministic automaton [15].

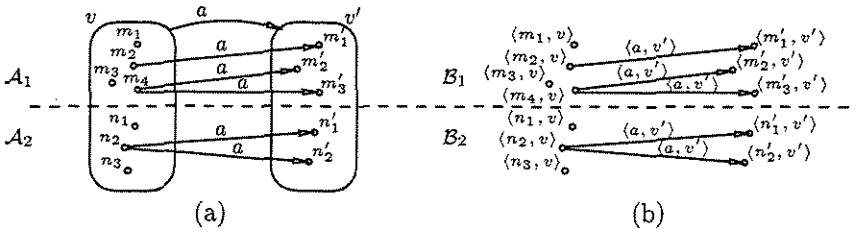

Figura B.2: The idea of a nondeterministic image

A run r of $\mathcal{B}$, over an ω-word $\sigma = \sigma_1\sigma_2\cdots$, is an infinite sequence $r_0r_1r_2\cdots$ of states, such that $r_0 \in Q_0$, and for all $i \geq 1$, $\langle r_{i-1}, r_i, \sigma_i\rangle \in T$. A run r is said to be *accepting* iff, for infinitely many $i \geq 1$, $r_i \in F$. The automaton $\mathcal{B}$ accepts an ω-word σ iff there is an accepting run of $\mathcal{B}$ over σ. The set $L(\mathcal{B})$ of ω-words accepted by $\mathcal{B}$ is its *language.*

The Nondeterministic Image.

The nondeterministic image encodes the runs of G in the alphabet of the ω-automata. Let $P = \{m \,|\, m \in P_v \text{ for some } \langle v, -\rangle \in V\}$ be the set of all positions of $\mathcal{A}_1$ in G. Then, $\mathcal{B}_1 = \langle \Delta, Q_1, Q_{0_1}, T_1, F_1\rangle$ and $\mathcal{B}_2 = \langle \Delta, Q_2, Q_{0_2}, T_2, F_2\rangle$, where $\Delta = \{\Sigma \cup \{\triangleright\}\} \times V$. For $\mathcal{B}_1$, we have the following definitions: $Q_1 = V \times P$; $Q_{0_1} = \{\langle\langle v_0, 1\rangle, m\rangle \,|\, m \in P_{v_0}\}$; $T_1 = \{\langle\langle\langle v, i\rangle, m\rangle, \langle\langle v', j\rangle, m'\rangle, \langle\sigma, \langle v', j\rangle\rangle\rangle \,|\, m \in P_v,\ m' \in P_{v'},$ there is an edge from $\langle v, i\rangle$ to $\langle v', j\rangle$ with label σ, and m is σ-linked to $m'\}$; $F_1 = \{\langle\langle v, 1\rangle, m\rangle \,|\, m \text{ is } \mathcal{A}_1\text{-accepting}\}$. For $\mathcal{B}_2$, the same definitions hold, changing S for P, $\mathcal{A}_2$ for $\mathcal{A}_1$, and S_v for P_v. See Fig. B.2 for the idea of the image, where part (a) represents the graph G and part (b) the obtained ω-automata. Then, the following theorem holds. Its proof is based on the correspondence between runs of $\mathcal{A}_1$ and $\mathcal{A}_2$, and runs of G; and on the correspondence between runs of G and runs of $\mathcal{B}_1$ and $\mathcal{B}_2$:

Theorem 4 *$L(\mathcal{A}_1) \subseteq L(\mathcal{A}_2)$ iff $L(\mathcal{B}_1) \subseteq L(\mathcal{B}_2)$.* □

Non-progressive $\mathcal{A}_1$.

In this case, there may be a run r of G, such that $|R_r| = 0$. The definition of acceptance for runs of G need to be changed. Now, a run $r = \langle v_0, 1\rangle \xrightarrow{\sigma_1} \langle v_1, 2\rangle \xrightarrow{\sigma_2} \langle v_2, 1\rangle \xrightarrow{\sigma_3} \cdots$ of G is $\mathcal{A}_1$-accepting iff there is an $\mathcal{A}_1$-run $m_0m_1m_2\cdots$ of r, such that for infinitely many $i \geq 0$, m_i is $\mathcal{A}_1$-accepting; *and* for every clock $x \in X_1$, for infinitely many $i \geq 0$, $m_i = \langle q_i, \mu_i\rangle$

and μ_i maps x to $[0,0]$ or to $(\uparrow)$.

From the automaton $\mathcal{B}_1$, one can easily obtain an new automaton $\mathcal{C}_1$ that accounts for this new condition, in a standard way. If $X_1 = \{x_1, x_2, \ldots, x_n\}$, then $\mathcal{C}_1$ is made of $n+1$ copies of $\mathcal{B}_1$. The new state space is $V \times P \times \{1, 2, \ldots, n+1\}$. While reading an ω-word, the control starts in the first copy, and jumps to the second copy as soon as it gets in a state $\langle\langle v, 1\rangle, \langle q, \mu\rangle, 1\rangle$, where μ maps x_1 to $[0,0]$ or to $(\uparrow)$. This process repeats for every clock until the control reaches the last copy. Then it jumps back to the first copy, as soon as it gets in a state $\langle\langle v, 1\rangle, m, n+1\rangle$, where m is $\mathcal{A}_1$-accepting. The new set of final states is $\{\langle\langle v, 1\rangle, m, n+1\rangle \mid m \text{ is } \mathcal{A}_1\text{-accepting}\}$.

B.5 Effective Infinite-State ω-Automata

The semi-decision procedure presented above can be viewed as a reduction of the language inclusion problem for NTA to the language inclusion problem for nondeterministic effective infinite-state ω-automata [50]. An infinite-state ω-automaton $\mathcal{B} = \langle \Delta, Q, Q_0, T, F\rangle$ is *effective* if the sets Δ, Q, Q_0, T and F are all recursive. That is, the sets may be infinite, but they are enumerable, and there is a Turing Machine $\mathcal{M}_\mathcal{B}$ that takes as input four integers w, x, y and z, always halts, and accepts the input iff: $x \in \Delta$ if $w = 1$; $x \in Q$ if $w = 2$; $x \in Q_0$ if $w = 3$; $\langle x, y, z\rangle \in T$ if $w = 4$; and $x \in F$ if $w = 5$ [50]. The machine $\mathcal{M}_\mathcal{B}$ can itself be encoded as an integer $\#\mathcal{B}$, the *index* of $\mathcal{B}$.

Given any two NTA $\mathcal{A}_1$ and $\mathcal{A}_2$, we can easily derive, from the semi-decision procedure, two indexes $\#\mathcal{B}_1$ and $\#\mathcal{B}_2$, for two nondeterministic effective infinite-state ω-automata $\mathcal{B}_1$ and $\mathcal{B}_2$, respectively, such that $L(\mathcal{A}_1) \subseteq L(\mathcal{A}_2)$ iff $L(\mathcal{B}_1) \subseteq L(\mathcal{B}_2)$. This shows[4] that the language inclusion problem and the universality problem for NTA are in Π^1_2, since the language inclusion problem for nondeterministic effective infinite-state ω-automata is known to be Π^1_2-complete [50] (we refer the reader to [46] for an introduction to the analytical hierarchy). The current complexity lower bound for the language inclusion problem and the universality problem for NTA is Π^1_1-hard [4]. The exact position of these undecidable problems in the analytical hierarchy is an interesting open problem.

[4]This result can also be shown directly by a Π^1_2 predicate form once we remind that for every timed word we can find another timed word with rational occurrence times [4], which is equivalent for the purpose of language inclusion; such that timed words can be "paired" with functions from $\mathbb{N}$ to $\mathbb{N}$.

B.6 Conclusions

In this paper, we presented a generalization of the region graph for NTA, which leads to a semi-decision procedure for testing language inclusion of NTA. We showed that TA models of real-time systems, satisfying a progress requirement, can be tested against any NTA. Interestingly enough, the semi-decision procedure is also a reduction of the language inclusion problem for NTA to the language inclusion problem for nondeterministic effective infinite-state ω-automata.

The method is expensive, as one should expect for the presence of nondeterminism in the specification. However, from a practical point of view, there is no difference between the previous algorithms for testing language inclusion for deterministic specifications and the procedure presented here (or, for that matter, the semi-decision procedure for reachability of hybrid automata [23]). Thus, one direction for future work is the development of heuristic methods and symbolic techniques that might expedite a useful implementation for the language inclusion problem.

Another direction is the interesting theoretical question about the exact position of the language inclusion problem and the universality problem for NTA in the analytical hierarchy. They are Π^1_1-hard [4], and belong to Π^1_2 (as a corollary of Section B.5).

Bibliografia

[1] R. Alur, C. Courcoubetis, e D. Dill. Model-checking for probabilistic real-time systems. Em *ICALP'91*, volume 510 de *LNCS*, páginas 115–136. Springer-Verlag, 1991.

Página(s) 76

[2] R. Alur, C. Courcoubetis, e D. Dill. Verifying automata specifications of probabilistic real-time systems. Em *Real-Time: Theory in Practice*, volume 600 de *LNCS*, páginas 28–44. Springer-Verlag, 1992.

Página(s) 75, 76, 77, 82, 84, 85, 86, 88, 94, 98, 99, 102

[3] R. Alur e D. Dill. Automata for modeling real-time systems. Em *ICALP'90*, volume 443 de *LNCS*, páginas 322–355, 1990.

Página(s) 42

[4] R. Alur e D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

Página(s) 1, 4, 6, 9, 13, 15, 20, 22, 23, 34, 35, 39, 42, 43, 45, 73, 76, 77, 80, 82, 83, 86, 88, 95, 96, 106, 108, 109, 110, 111, 112, 114, 116, 117, 119, 120

[5] R. Alur, L. Fix, e T. Henzinger. Event-clock automata: A determinizable class of timed automata. Em *CAV'94*, volume 818 de *LNCS*, páginas 1–13, 1994.

Página(s) 4, 97, 106

[6] R. Alur e T. Henzinger. Time for logic. *SIGACT News*, 22(3):6–12, 1991.

Página(s) 73

[7] R. Alur e T. Henzinger. Logics and models of real-time: A survey. Em *Real-Time: Theory and Practice*, volume 600 de *LNCS*, páginas 74–106, 1992.

Página(s) 2, 73, 76

[8] E. Asarin e O. Maler. Achilles and the tortoise climbing up the arithmetical hierarchy. *J. of Comp. and Sys. Sci.*, 57:389–398, 1998.

Página(s) 10

[9] E. Asarin, O. Maler, e A. Pnueli. On discretization of delays in timed automata and digital circuits. Em *CONCUR'98*, volume 1466 de *LNCS*, páginas 470–484, 1998.

Página(s) 2, 97, 103, 106, 114, 115

[10] D. Beauquier. Pumping lemmas for timed automata. Em *FoSSaCS*, volume 1378 de *LNCS*, páginas 81–94. Springer-Verlag, 1998.

Página(s) 10

[11] B. Berard, V. Diekert, P. Gastin, e A. Petit. Characterization of the expressive power of silent transitions in timed automata. *Fundamentae Informaticae*, 36(2):145–182, 1998.

Página(s) 10

[12] O. Bournez. Achilles and the tortoise climbing up the hyper-arithmetical hierarchy. *Theoretical Computer Science*, 210:21–71, 1999.

Página(s) 10

[13] P. Bouyer e A. Petit. Decomposition and composition of timed automata. Em *ICALP'99*, volume 1466 de *LNCS*, páginas 210–219, 1999.

Página(s) 10, 106

[14] J. Castro e F. Cucker. Nondeterministic ω-computations and the analytical hierarchy. *Zeitschr. f. math. Logik und Grundlagen d. Math.*, 35:333–342, 1989.

Página(s) 8, 25

[15] Y. Choueka. Theories of automata on ω-tapes: A simplified approach. *J. of Comp. and Sys. Sci.*, 8:117–141, 1974.

Página(s) 1, 3, 117

[16] C. Courcoubetis e M. Yannakakis. The complexity of probabilistic verification. *Journal of the ACM*, 42(4):857–907, 1995.

Página(s) 33, 77, 98, 99, 101, 102

[17] A. Göllü, A. Puri, e P. Varaiya. Discretization of timed automata. Em *Anais da 33rd IEEE Conference on Decision and Control*, páginas 957–958, 1994.

Página(s) 2, 86, 88

[18] N. Halbwachs, Y. Proy, e P. Raymond. Verification of linear hybrid systems by means of convex approximations. Em *Int. Symposium on Static Analysis*, volume 864 de *LNCS*, 1994.

Página(s) 103

[19] D. Harel, A. Pnueli, e J. Stavi. Propositional dynamic logic of nonregular programs. *J. of Comp. and Sys. Sci.*, 26:222–243, 1983.

Página(s) 22, 57

[20] F. Hennie. *Introduction to Computability.* Addison Wesley, 1977.

[21] T. Henzinger. Sooner is safer than later. *Information Processing Letters*, 43:135–141, 1992.

Página(s) 62

[22] T. Henzinger. *The Theory of Hybrid Automata*, volume 170 de *NATO ASI Series F: Computer and Systems Sciences, Verification of Digital and Hybrid Systems*, páginas 265–292. Springer-Verlag, 2000.

Página(s) 6

[23] T. Henzinger, P.-H. Ho, e H. Wong-Toi. Hytech: the next generation. Em *Anais do 16th Annual IEEE Real-Time Systems Symposium*, páginas 56–65, 1995.

Página(s) 6, 103, 120

[24] T. Henzinger, P. Kopke, A. Puri, e P. Varaiya. What's decidable about hybrid automata? *Journal of Computer and System Sciences*, 57:94–124, 1998.

Página(s) 24

[25] T. Henzinger, P. Kopke, e H. Wong-Toi. The expressive power of clocks. Em *ICALP 1995*, volume 944, páginas 417–428, 1995. LNCS.

Página(s) 62

[26] T. Henzinger, Z. Manna, e A. Pnueli. What good are digital clocks? Em *ICALP'92*, volume 623 de *LNCS*, páginas 545–558. Springer-Verlag, 1992.

Página(s) 2

[27] T. Henzinger e J. Raskin. Robust undecidability of timed and hybrid systems. Relatório Técnico USB/CSD-99-1073, Berkeley Univ., Outubro 1999.

Página(s) 96, 114

[28] T. Henzinger, J. Raskin, e P. Schobbens. The regular real-time languages. Em *ICALP'98*, volume 1443 de *LNCS*, páginas 580–591, 1998.

Página(s) 10, 76, 97

[29] T. Henzinger e J.-F. Raskin. Robust undecidability of timed systems. Em *HSCC'2000*, volume 1790 de *LNCS*, páginas 145–159. Springer-Verlag, 2000.

Página(s) 62

[30] P. Herrmann. Timed automata and recognizability. *Information Processing Letters*, 65:313–318, 1998.

Página(s) 34

[31] P. Hinman. *Recursion-Theoretic Hierarchies*. Springer-Verlag, 1978.

Página(s) 17

[32] T. Hirst e D. Harel. Taking it to the limit: On infinite variants of NP-complete problems. *J. of Comp. and Sys. Sci.*, 53:180–193, 1996.

Página(s) 17

[33] M. Kwiatkowska, G. Norman, R. Segala, e J. Sproston. Verifying quantitative properties of continuous probabilistic real-time graphs. Relatório Técnico CSR-98-11, Univ. of Birmingham, 1998.

Página(s) 76, 83

[34] H. Lewis. Finite-state analysis of asynchronous circuits with bounded temporal uncertainty. Relatório Técnico TR-15-89, Harvard Univ., 1989.

Página(s) 2, 97, 103, 115

[35] O. Maler e A. Pnueli. Timing analysis of asynchronous circuits using timed automata. Em *CHARME'95*, volume 987 de *LNCS*, páginas 189–205, 1995.

Página(s) 97, 103, 106, 114

[36] J. Miller. Decidability and complexity results for timed automata and semi-linear hybrid automata. Em *HSCC'2000*, volume 1790, páginas 296–309. Springer-Verlag, 2000. LNCS.

Página(s) 10

[37] A. Moura e G. Pinto. On the verification of nondeterministic automata specifications of probabilistic real-time systems. Relatório Técnico IC-99-27, Instituto de Computação, UNICAMP, Dezembro 1999.

[38] A. Moura e G. Pinto. On the verification of nondeterministic automata specifications of probabilistic real-time systems. Em *3rd Workshop on Formal Methods - SBC*, páginas 181–192, João Pessoa, Brazil, 2000.

[39] A. Moura e G. Pinto. A semi-decision procedure for testing language inclusion of nondeterministic timed automata. Relatório Técnico IC-00-02, Instituto de Computação, UNICAMP, Janeiro 2000.

[40] A. Moura e G. Pinto. Classes of timed automata and the undecidability of universality. Relatório Técnico IC-01-20, Instituto de Computação, UNICAMP, Dezembro 2001.

[41] A. Moura e G. Pinto. On almost deterministic timed automata. Relatório Técnico IC-01-06, Instituto de Computação, UNICAMP, Maio 2001.

[42] A. Moura e G. Pinto. Classes of timed automata and the undecidability of universality. Em *Anais do TPTS - Workshop of Theory and Practice of Timed Systems*, volume 65(6) de *ENTCS - Electronic Notes in Theoretical Computer Science*, Grenoble, France, Abril 2002. Elsevier Science.

Página(s) 9

[43] A. Moura e G. Pinto. A note on the verification of automata specifications of probabilistic real-time systems. *Information Processing Letters*, 82(5):223–228, Junho 2002.

Página(s) 74

[44] P. Odifreddi. *Classical Recursion Theory*, volume I. North-Holland, 1989.

Página(s) 17

[45] J. Ouaknine e J. Worrell. Universality and language inclusion for open and closed timed automata. Em *HSCC'2003*, volume 2623 de *LNCS*, páginas 375–388, 2003.

Página(s) 4

[46] H. Rogers. *Theory of Recursive Functions and Effective Computability.* The MIT Press, 1987.

Página(s) 7, 9, 10, 17, 18, 21, 119

[47] S. Safra. On the complexity of ω-automata. Em *Proc. of the 29th IEEE Foundations of Computer Science*, páginas 319–327, Outubro 1988.

Página(s) 8, 26, 33, 43, 117

[48] S. Safra. *Complexity of Automata on Infinite Objects.* PhD thesis, The Weizmann Institute of Science, Rehovot, Israel, 1989.

Página(s) 2, 26

[49] G. Shedler. *Regeneration and Network of Queues.* Springer-Verlag, 1987.

Página(s) 78

[50] A. P. Sistla. On verifying that a concurrent program satisfies a nondeterministic specification. *Information Proc. Letters*, 32:17–23, 1989.

Página(s) 8, 25, 107, 119

[51] T. Wilke. Specifying timed state sequences in powerful decidable logics and timed automata. Em *FTRTFT'94*, volume 863 de *LNCS*, páginas 694–715. Springer-Verlag, 1994.

Página(s) 2, 106, 115